AIRLINE CREW PAIRING OPTIMIZATION PROBLEMS AND CAPACITATED VEHICLE ROUTING PROBLEMS

A Thesis
Presented to
The Academic Faculty

by

Shengli Qiu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2013

AIRLINE CREW PAIRING OPTIMIZATION PROBLEMS AND CAPACITATED VEHICLE ROUTING PROBLEMS

Approved by:

Dr. Ellis L. Johnson,
Committee Chair
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. George L. Nemhauser
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Ellis L. Johnson, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Joel S. Sokol
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. William J. Cook
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Barry C. Smith
Barry C Smith LLC

Date Approved: June 2012

ACKNOWLEDGEMENTS

First and foremost I would like to express my sincere and deep gratitude to my advisor, Dr. Ellis Johnson, for his insightful guidance and motivation throughout this research, and his encouragement, patience and support that makes possible the completion of this dissertation. Dr. Johnson introduced me to the field of Airline Operations Research and gave me the opportunity to work on many interesting research problems.

I would like to thank Dr. William Cook, Dr. George Nemhauser, Dr. Joel Sokol, and Dr. Barry Smith for taking time out of their busy schedules to serve on my dissertation committee and give me constructive comments and suggestions. I would also extend my thanks to Dr. Earl Barnes and many other faculty at ISyE from whom I have learned so much valuable knowledge.

I would like to extend my global thanks to the staff at Georgia Tech from whom I have received all kinds of help. I also thank my fellow students who make my school experience there so special.

I am also so grateful to my family for their enduring patience and constant love, encouragement and confidence that helps me so much through the journey of my Ph.D. study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Crew pairing and vehicle routing are combinatorial optimization problems that have been studied for many years by researchers worldwide. The aim of this research work is to investigate effective methods for solving large scale crew pairing problems and vehicle routing problems.

In the airline industry, to address the complex nature of crew pairing problems, we propose a duty tree method followed by a primal-dual subproblem simplex method. The duty tree approach captures the constraints that apply to crew pairings and generate candidate pairings taking advantage of various proposed strategies. A huge number of legal pairings are stored in the duty tree and can be enumerated. A set partitioning formulation is then constructed, and the problem is solved using a primal-dual subproblem simplex method tailored to the duty tree approach. Computational experiments are conducted to show the effectiveness of the methods.

We also present our efforts addressing the capacitated vehicle routing problem (CVRP) that is the basic version of many other variants of the problem. We do not attempt to solve the CVRP instances that have been solved to optimality. Instead, we focus on investigating good solutions for large CVRP instances, with particular emphasis on those benchmark problems from the public online library that have not yet been solved to optimality by other researchers and determine whether we can find new best-known solutions. In this research, we propose a route network that can store a huge number of routes with all routes being legal, a set partitioning formulation that can handle many columns, and the primal-dual subproblem simplex method to find a solution. The computational results show that our proposed methods can achieve better solutions than the existing best-known solutions for some difficult instances.

Upon convergence of the primal-dual subproblem simplex method on the giant-tour based networks, we use the near optimal primal and dual solution as well as solve the elementary shortest path problem with resource constraints to achieve the linear programming relaxation global optimal solution.

CHAPTER I

INTRODUCTION

1.1 The Airline Crew Scheduling Problem

The airline industry is known to have extremely valuable assets, high labor costs, interdependent resources, complex operations involving security and safety concerns, and competitive pressures. To manage large-scale operations, improve decisions, and increase profits, the airline industry has become a hotbed for development and application of operations research methods.

Due to the large operational scale, the airline planning problem is often decomposed into smaller subproblems to decrease problem size and enhance tractability. The subproblems roughly include flight schedule generation, fleet assignment, aircraft maintenance routing, and crew scheduling. These planning steps are usually performed in sequence with the output from upstream steps providing input to downstream steps, with different practices showing wide variety in the detailed planning procedures involved.

Flight Schedule Generation: The main goal of flight schedule generation is to determine the origin-destination itineraries, frequencies, and flight times in a given time horizon, based on the forecasted demand, available resources, and so on.

Fleet Assignment: The fleet assignment problem is determining which type of aircraft should be assigned to each flight. The objective is to match the demand and capacity as closely as possible, taking into account the availability of aircraft in each fleet.

Aircraft Maintenance Routing: The maintenance routing problem concerns assigning individual aircraft to flights to ensure that the maintenance check requirement is satisfied. The main objective is to determine routing that starts and ends at

the same station for each aircraft in a fleet, subject to the condition that the aircraft be at certain maintenance stations at preplanned intervals.

Crew Scheduling: Airline crew scheduling is an important part of airline operations and profitability because crew costs comprise the second largest operating expense for airlines and small improvements in efficiency can yield large financial benefits. Due to its complexity, the crew scheduling problem is usually solved through two major sequential steps of crew pairing and crew rostering.

Crew Pairing: Crew pairing describes a sequence of flights with the start and end of the sequence being the same crew base airport. A good pairing not only needs to meet the coverage requirement of each flight but also must satisfy many government, labor union, and airline operational rules. Though specific objectives of various crew pairing problems may differ, the general objective is to minimize pairing costs, which is a nonlinear function of many components involved. The obvious combinatorial explosion that would occur makes the problem a rather large one and, thus, difficult to solve.

Crew Rostering: The main focus of crew rostering is assigning anonymous pairings to individual crew members to produce personalized rosters. The goal of solving the crew rostering problem is to find assignments that maintain the quality of life for the crew while keeping the costs in mind. The optimization problem is solved to select exactly one roster for each crew member. Similar to crew pairing, crew rostering also has to meet complex rules and regulations imposed by airlines, labor unions, and government agencies.

Recently, interest in integrating crew pairing and crew rostering into one planning problem is growing because an integrated formulation could produce better rosters in terms of both costs and quality of life. As mentioned by Kohl and Karisch [77], some researchers consider integrating these problems an important research area in crew planning; however, the size of the integrated problem makes it even more challenging

to solve.

Johnson [70] presented a good summary of optimization in airline scheduling, including successes, challenges, and new directions. For another discussion of the accomplishments and opportunities in the airline scheduling planning, we refer the reader to the work by Barnhart and Cohn [19].

1.2 The Vehicle Routing Problem

The vehicle routing problem (VRP) is one of the most studied combinatorial optimization problems. It is focused on determining an optimal set of routes for a fleet of vehicles to serve a given set of customers [110]. Since first being introduced by Dantzig and Ramser [41], the VRP has received great attention due to its practical relevance and its considerable difficulty. Various models and algorithms have been proposed to find the optimal or approximate solutions for the different variants of the VRP. In the VRP family, the capacitated VRP, the distance-constrained VRP, the VRP with time windows, the VRP with backhauls, and the VRP with pickup and delivery, play as major members. The capacitated VRP imposes a vehicle capacity restriction, and the distance-constrained VRP has a constraint on the maximum length or maximum time for each route. In the VRP with time windows, besides the capacity constraint, each customer to be served is associated with a time interval. The VRP with backhauls has two subsets of customers, one needing the delivery service and the other needing the pickup service. Finally in the VRP with pickup and delivery, each customer needs both delivery and pickup service.

In the VRP, the main components usually involved are the road network, customers, depots, vehicles, and drivers. When modeling, different operational constraints and typical optimization objectives are taken into consideration for the particular VRP. In the literature, several different basic formulation approaches for the VRP are proposed. The first type is the vehicle flow formulation, which uses integer

variables to model the number of times the arc or edge of the constructed graph is traversed by a vehicle. The second type is the commodity flow formulation, which considers the flow of the commodities along the vehicle paths. The third type is the set partitioning formulation, which uses binary variables for different feasible routes. These models have their different advantages and applicability. For more detailed discussion of the definitions, variants, and exact and heuristic approaches for the VRP, we refer the interested reader to the book edited by Toth and Vigo [110].

1.3 Research Purpose

The airline crew pairing optimization problem is an essential part of the crew scheduling problem. It has been studied for many years and is still very challenging. One of the main goals of this research is to contribute to solving large airline crew pairing problems.

Another intensely researched topic in combinatorial optimization is the VRP. In this work, we focus on the capacitated vehicle routing problem (CVRP). According to the online VRP library, CVRP instances with up to 134 customers have been solved to optimality by researchers, but several larger cases have not been solved to optimality, such as M-n151-k12 (150 customers), M-n200-k16 (199 customers), M-n200-k17 (199 customers), and G-n262-k25 (261 customers). We are interested in investigating whether we can find new best-known solutions to these problems. In addition, we seek a higher quality solution for large-scale CVRP instances and try to determine whether our approach can work with existing heuristic approaches to improve the quality of their solutions, whether route-first, cluster-second methods are competitive, and whether the set partitioning formulation is effective for large CVRP instances.

1.4 Thesis Outline

This thesis is organized into five chapters. Following the introduction are two chapters discussing our methods for solving the crew pairing optimization problem and one chapter investigating the methods for solving CVRP and the CVRP on trees. The detailed chapter structures are as follows.

Chapter I includes basic introductions to topics, discussion of the research purpose, and the research topics to be studied: the airline crew pairing optimization problem and the capacitated vehicle routing problem.

Chapter II largely consists of discussion of the crew pairing optimization problem. In particular, a duty tree method with a compact storage scheme is proposed to generate pairings. The legality check for crew pairings are done once for all. With the proposed strategies, the memory requirement during the whole pairing generation process is significantly much smaller than that required by traditional approaches, as can be seen from the computational runs.

Chapter III is devoted to the method for solving the crew pairing optimization problem. In this chapter, we formulate the crew pairing problem as a set partitioning problem and solve it using a primal-dual subproblem simplex algorithm based on the duty tree method. Some computational results are also provided.

Chapter IV includes discussion of our research efforts focused on a primary topic among combinatorial optimization problems, the vehicle routing problem. In that chapter, we focus on the CVRP that plays as a basic version of the problem family. We propose an approach that generally includes route network construction, the set partitioning formulation, and the primal-dual subproblem simplex solution method in an attempt to achieve better solutions than previously presented by other researchers. Some computational runs are performed using the public library instances, and the detailed results are provided.

In addition, Chapter IV includes discussion of applying our CVRP approaches to

a variant of the capacitated vehicle routing problem, that is, the capacitated vehicle routing problem on trees, which arises in some real applications. Finally, some computational experiments are conducted with the expectation of solving much larger cases than those reported in literature.

Chapter V concludes this thesis by providing a summary of the work and suggestions for further research.

CHAPTER II

AIRLINE CREW PAIRING USING DUTY TREE

2.1 Introduction

In the airline industry, after developing the flight schedule and assigning aircraft to cover all the flight legs in the schedule, one important next step is to construct crew schedules for these flights. Because crew cost is second only to fuel cost in airline operations, airlines devote great effort to achieving efficient crew utilization. Thus, how to build good crew schedules has been studied intensively. An important step is to generate pairings that can cover all required flights and minimize excess cost. The crew scheduling problem is usually solved as two sequential subproblems: crew pairing and crew rostering (see Figure 2.1). The necessary inputs, including the flight scheduling, fleet assignment, and aircraft maintenance routing, are usually obtained during the upstream planning.

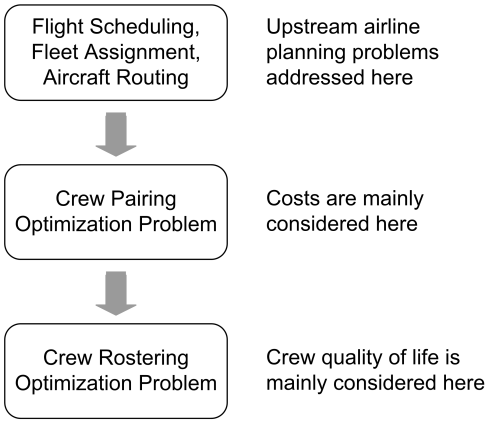

Figure 2.1: The crew scheduling optimization problem.

The two subproblems usually have different focuses, with the pairing problem being focused more on costs and the rostering problem addressing crew quality of life. In a crew pairing problem, sequences of flights starting and ending at a crew base are

generated and selected for subsequent assignment of crews while the rostering problem is solved with pairing solutions as inputs while taking into account crew members' individual needs, such as training, days off, and other preferences. Crew pairings are subject to various constraints, such as government safety regulations, company operational rules, and labor union contract terms. In addition, pairing costs are usually nonlinear and associated with many aspects of pairings, for example, salary, per diem compensation, hotel costs, and so forth. The general objective of the crew pairing optimization problem is to minimize pairing costs while satisfying various constraints. Crew pairing is an interesting and challenging research problem with its large-scale combinatorial explosive nature in addition to the complex rules and nonlinear costs. In this research, we are interested in investigating approaches that can contribute to solve very large airline crew pairing optimization problems.

2.2 Basic Definitions

In crew pairing, a *flight leg* is a nonstop flight, also referred to as a *segment*. A sequence of flights that can be flown by a single crew in a work day is often referred to as a *duty period* or *duty* in short. Duties are subject to many airline and government rules. For example, flights must be sequential in terms of space and time in a duty, and the sit time or connection time between two sequential flights must be above a minimum and below a specified maximum. In addition, regulations address, among other considerations, total flying time or block time, and maximum elapsed time in a duty. The crew cost associated with a duty includes several aspects, for example, crew's actual flying time, the total elapsed time of the duty, and the guaranteed time for the duty. The duty cost is usually expressed in terms of time as follows:

$$c_d = \max\{\sum block, f_d * elapse, min\ guar\}$$

,

where f_d is a factor that can be defined based on different applications.

A *crew pairing* is a sequence of flights that starts and ends at a crew base. A pairing usually consists of duties with overnight rests or layovers in between. If the layover period lasts longer than 24 hours, it is called *double overnight*. A pairing usually lasts two or three days and a crew usually works four or five pairings in a month. Sometimes, *deadheading* is necessary to reposition a crew for the next flight by flying as a passenger. A pairing example is illustrated in Figure 2.2.

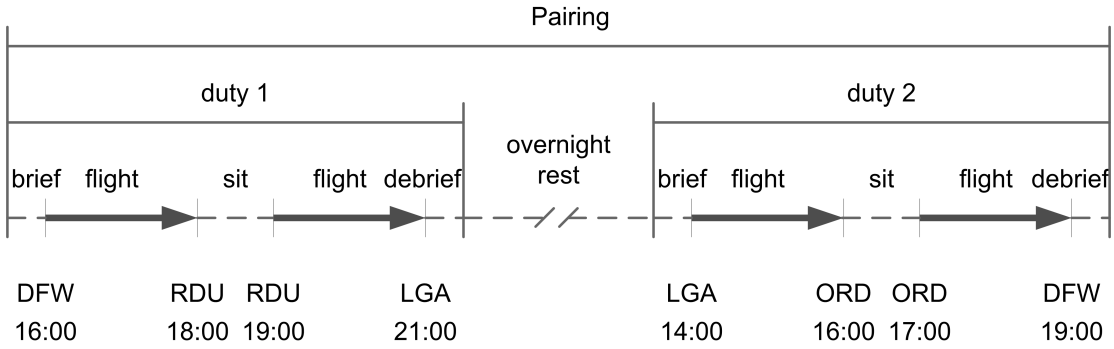

Figure 2.2: An example of crew pairing with DFW airport as crew base.

The pairing in Figure 2.2 is based at Dallas-Fort Worth (DFW) airport. It has two duties, with the first starting from DFW at 16:00, flying to Raleigh-Durham (RDU), then to LaGuardia (LGA), and taking overnight rest there, and the second starting from LGA, flying to Chicago OHare (ORD), and flying back to DFW at 19:00 the next day. In this example, the times are rounded to hours, and all are considered to be in the same time zone [8]. Each duty begins with a short briefing session and ends with a short debriefing session for crews to sign in, sign out, and handle any necessary paperwork. In the illustrated pairing, there is a sit time between flights in each duty, and a layover or overnight rest between duties.

Pairings are also constrained by many requirements, for example, the maximum number of duties in a pairing, the minimum and maximum rest time between duties,

and the maximum elapsed time of a pairing, which is often referred to as *time-away-from-base* (TAFB). In the U.S., a particularly complicated constraint in constructing pairings is the 8-in-24 rule imposed by the Federal Aviation Agency (FAA). The 8-in-24 rule specifies a longer layover requirement when the flying time is more than eight hours in any 24-hour period.

The pairing cost is complex and includes such components as duty costs, TAFB, guaranteed time, and extra costs for meals and hotels. It is usually expressed as

$$c_p = \max\{\sum_{d \in p} c_d, f_p * TAFB, n_d * avg\ duty\ guar\} + \sum_{\hat{d} \in p, \bar{d} \in p, \hat{d} \to \bar{d}} e(\hat{d}, \bar{d})$$

,

where f_p is a factor that can be defined based on different applications; n_d is the number of duties in pairing p; d, $\hat{d}$, and $\bar{d}$ are the duties in p; $\hat{d} \to \bar{d}$ indicates that duty $\hat{d}$ is immediately followed by duty $\bar{d}$ in p, and $e(\hat{d}, \bar{d})$ is the extra cost associated with the rest between $\hat{d}$ and $\bar{d}$ [20].

To minimize the expense beyond the cost of actual flying time, a *pay-and-credit* term is often used in the U.S., which is calculated based on the following equation:

$$pay\text{-}and\text{-}credit = \frac{pairing\ cost - \sum block\ time}{\sum block\ time}$$

Generally, there are three types of crew pairing problems: daily problems, weekly problems, and dated problems. The daily problem builds pairings based on the set of flights assumed to fly every day while the weekly problem is for flights repeated weekly and the dated problem concerns specific days of a month.

2.3 Literature Review

The crew pairing problem has been well studied over the years. As early as the 1960s, a survey on this topic was published by Arabeyre et al. [10]. Other early works include Rubin [99], Marsten and Shepardson [84], Bornemann [28], Baker et al. [12],

Ball and Roberts [17], Etschmaier and Mathaisel [49], Gerschkoff [53], Anbil et al. [7], and Graves et al. [57]. Some researchers tried to apply a column generation method for crew pairing. For example, Minoux [87] tried to solve the linear relaxation of the crew pairing problem using column generation, and Lavoie et al. [80] discussed further extensions. The column generation approach was also discussed by Crainic and Rousseau [38] and Barnhart et al.[18] while integrating the column generation into a branch-and-bound approach was tried by Desaulniers et al. [42]. In addition, a branch-and-cut approach was discussed by Hoffman and Padberg [62]. Furthermore, Chu et al. [31] and Wedelin [112] focused on optimization algorithms for underlying crew scheduling mixed integer programs, and a similar effort with a different formulation was discussed by Vance et al. [111]. AhmadBeygi et al. [4] also presented an integer programming approach to generate crew pairings.

Extensions of the crew pairing problem dealing with different aspects have also been studied. Barnhart et al. [21] discussed crew deadheading to improve crew pairing solutions through efficient selection and use of deadhead flights. Schaefer et al. [103] presented an evaluation of crew schedule quality to find crew schedules that yielded better performance despite disruptions. Similar research efforts have generated robust plans that are less sensitive to disruptions arising in the execution of the planned schedule [48], [98], [105], [115]. In addition, Klabjan et al. [75] developed a model to capture regularity by adding a second goal of maximizing the repetition of itineraries over a weekly horizon. Klabjan et al. [76] introduced a random pairing generation routine into column selection during the solution process.

Usually, the connection time between two flight legs can be shorter if no crew transfer between aircraft is needed; that is, both flight legs are operated by the same aircraft, and same crew is assigned to cover both flights. Some researchers such as Cohn and Barnhart [34], Cordeau et al. [37], Klabjan et al. [74], Mercier [86], and Weide et al. [113] have tried to integrate aircraft routing and crew scheduling

problems to explore achieving good plans in terms of capturing both a cost-minimizing crew pairing solution and a feasible maintenance routing solution. Saddoune et al. [100] proposed an integrated model for the crew pairing and crew bidline problems and solved it using the combined column generation/dynamic constraint aggregation. The challenge in such integrated problems is the resulting problem size is even bigger and harder to solve for larger airlines.

Various heuristic techniques have also been tried by some researchers. For example, Broderick et al. [39] tried to solve the crew pairing problem using ant colony optimization and constraint programming. In addition, Housos and Elmroth [63] tried to solve the crew pairing problem over a week horizon using an iterative a-day-at-a-time scheme.

In many real world applications, the crew pairing problems are usually tackled with a three-phase approach that includes the daily problem, the weekly problem, and the monthly problem [73]. Saddoune et al. [101] proposed a rolling horizon heuristic approach in trying to outperform the three-phase approach. Finally, crew pairing problems differ somewhat for short haul and long haul problems. For example, the long haul networks are often relatively sparser than the short haul networks, and the flight schedule is often a weekly schedule [22].

For a specific review of the definitions, formulations, solution algorithms, and other elements of airline crew scheduling problems, we refer the reader to the work by Barnhart et al. [20] and Gopalakrishnan and Johnson [56]. Additional topics beyond crew scheduling, including customer modeling, airline planning and schedule development, revenue management, air travel distribution, airline operations, and air traffic flow management, are covered in the book [23].

With the development of operations research theory as well as computing software and hardware, the application of optimization tools to crew scheduling problems has been long accepted by airlines [33]. As a result, the cost of crew pairing solutions

exceeding flying costs has been reduced from 10-15% to 1-2%, usually translating to millions of dollars savings annually for large airlines.

Considering the combinatorial explosive nature of the number of crew pairings and the airlines achieving more efficient and flexible crew utilization by integrating resources, billions, trillions, or even many more columns in the problem can be expected to be solved. In addition, airline operations are highly complicated processes with many expensive resources, such as fuels, crews, aircrafts, airports, and maintenance facilities. In recent years, the pressure on the airline industry has further intensified due to increasing competition, rising fuel costs, more congestion and security concerns. This situation makes attractive simultaneous modeling and solving of different subproblems currently solved sequentially while bringing challenges of far larger problems.

The ability to generate crew pairings effectively is crucial in solving airline crew scheduling problems, whether with robust planning or integrated planning. Though significant progress has been made in crew scheduling, challenges still remain. Details of solving big crew pairing optimization problems will be described in subsequent sections and solution methods in the next chapter.

2.4 The Flight Network and the Duty Network

In the airline industry, two types of networks are usually used in solving optimization problems. One is the flight-based network, and the other is the duty-based network.

The flight network represents each flight with two distinct nodes, the departure node and the arrival node, and an arc connecting the two nodes. If the arrival station of the first flight is the same as the departure station of the second flight, and the connection time is between the minimum sit time and maximum sit time, then an arc is used to connect the arrival node of the first flight and the departure node of the second flight. Let $G = (N, A)$ be the time-space flight network where N is the

node set and A is the arc set. The nodes N represent the departure and the arrival of flight legs as well as the source s and the sink t. For daily problems, each flight arc can be replicated as many times as the maximum number of calendar days allowed in a pairing. The source node s is connected to the departure node of each flight departing from a specified crew base, and the sink node t is connected by the arrival node of each flight arriveing at that crew base. The legalities associated with duties and pairings need to be handled during pairing generation.

The duty network builds nodes and arcs in a similar manner. Usually, the nodes represent the departure time and airport, and arrival time and airport of the duties. The arcs represent duties, and the connection arcs are used for the legal rest connections. That is, the connection times are required to be within the minimum rest time and the maximum rest time. The duties are enumerated beforehand, so the duty-related legality rules are already captured.

Usually the flight network and the duty network have different orders of magnitude in the number of nodes and arcs. The duty network requires more storage space.

A partial flight network for the following flights is illustrated in Figure 2.3.

Flight 1: airport A - airport B 08:00 - 09:00

Flight 2: airport B - airport C 10:00 - 11:00

Flight 3: airport C - airport D 13:00 - 14:00

Flight 4: airport D - airport A 15:00 - 16:00

The network spans a two-day time horizon and includes two copies of each flight, with the solid arcs representing flights and the dotted arcs representing possible connections between flights. Each arrival node has two connections from it, one to the next departure and the other to the same departing flight one day later [20].

Figure 2.4 shows the duty network. The nodes represent the departure and arrival for each duty period. The solid arcs represent duty periods while the dotted arcs

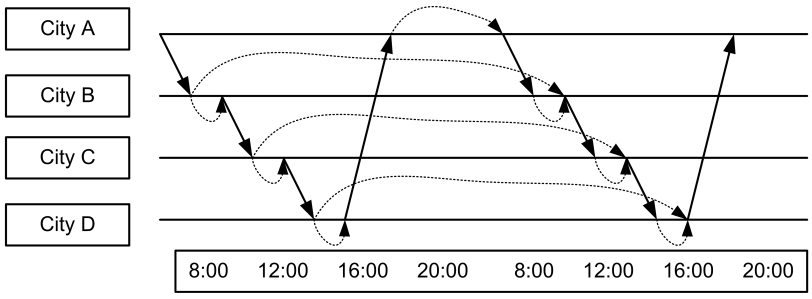

Figure 2.3: A flight network example [20].

represent connections between duties. The duty network captures the duty level legality rules.

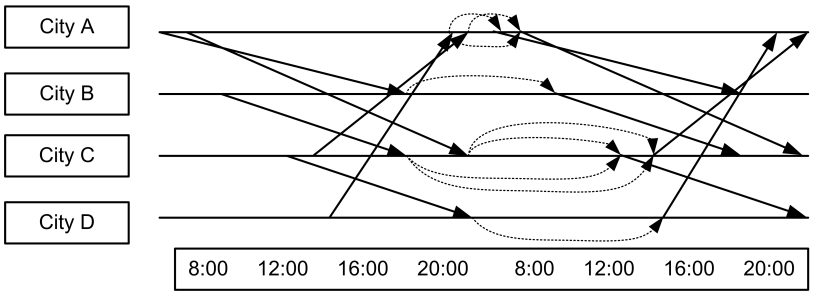

Figure 2.4: A duty network example [20].

In both flight and duty networks, a legal pairing is a $s - t$ path in the network. Although connection times can be kept legal according to the rules, a $s - t$ path still may not necessarily be a legal pairing due to many other pairing rules, for example, maximum number of duties, maximum time away from base, and so on. Usually, because of the complexity of rules involved in crew pairing optimization problems, many paths from the network fail to satisfy all rules, causing inefficiency in the approaches based on flight and duty networks. In this research, we propose a new approach.

2.5 The Duty Tree Method for Crew Pairing Optimization

The purpose of this research is to investigate an effective approach to solving bigger crew pairing optimization problems, either with pure enumeration methods or with enumeration embedded in column generation. First, the compact storage concept, the thread, is discussed.

2.5.1 Thread Compact Storage

With the explosive nature of the crew pairing problem, the number of possible connections within and between duties to form pairings usually imposes a memory difficulty during the pairing generation and optimization if pairings are explicitly stored. To address this difficulty, some compact data storage schemes have been developed. Hu [64] presented a pairing storage approach in which all the pairings are stored using some strings and a compact array that is much smaller in size. In this approach, all the pairings are first arranged in depth-first order and then written one by one in order into a one-dimensional array without duplicating the same initial part of any two adjacent pairings. A marker is used between any two adjacent pairings to identify how many to back track from the end of the first pairing and then how many to go forward to form the second pairing, and the string is used to represent the same series of flight legs. An application of this can be found in the work by Lettovsky et al. [81]. Further, based on Hu's work [64], Shaw [104] devised a compacting scheme on a string network and improved on it with better string choices and use of default markers. For more details about these compacting approaches, we refer the reader to [64] and [104].

In this research, we use a similar compact array concept and implement it in a thread logic. For example, 10 flights can be represented simply as $1, 2, 3, ..., 10$ based on their indices. The resulting 10 duties are shown in Figure 2.5.

The thread for the duties is as follows:

$Duty_1 = \{1\}$	$Duty_2 = \{1, 2\}$	$Duty_3 = \{1, 2, 3\}$	$Duty_4 = \{1, 2, 3, 4\}$
$Duty_5 = \{1, 2, 3, 5\}$	$Duty_6 = \{1, 6\}$	$Duty_7 = \{1, 6, 7\}$	$Duty_8 = \{1, 6, 8\}$
$Duty_9 = \{1, 6, 8, 9\}$	$Duty_{10} = \{1, 6, 8, 10\}$		

Figure 2.5: A list of duty periods.

$$\{1\ 2\ 3\ 4\ {-1}\ 5\ {-3}\ 6\ 7\ {-1}\ 8\ 9\ {-1}\ 10\ \}$$

In the thread, we add a marker to indicate how many to back track from the end of the previous duty, followed by the flights going forward to form the next duty. Thus, the thread is a one-dimensional array of integers with flight indices and markers. The thread indices define a traversal of a tree, that is, a sequence of nodes that walks its way through the nodes of a tree, starting from the root node and visiting nodes in an order as indicated in the indices. The thread indices follow a depth-first search order of the tree. In terms of size, the thread format is much smaller than the explicit list of duties. In this research, in addition to the thread logic used during implementation, a new duty tree scheme is also introduced.

2.5.2 The Duty Tree

In a pairing, each duty is represented by a list of flights in the order of departure time. In this research, these duties are stored in a tree structure, that is, the duty tree, which is a basic building block in this overall approach. It is assumed that each flight can occur only once in a pairing. The duty tree structure will be discussed using the following basic concepts.

Tree: A tree is a connected graph that contains no cycle [5]. For detailed tree properties, we refer the reader to [5] and [114]. In this duty tree, we allow a tree that contains only the root node. For example, a long haul flight from ORD to PVG is a legal duty itself, and no day connection is legal for it to construct a 2-flight duty.

This flight's duty tree can contain only the root node itself.

Rooted tree: A rooted tree is a tree with a specially designated node, called its root [5]. There are two special types of rooted trees: a directed in-tree and a directed out-tree [5].

Directed out-tree: A tree is a directed out-tree rooted at node s if the unique path in the tree from node s to every other node is a directed path [5].

The duty tree proposed in this study is a directed out-tree and is defined in Definition 2.5.1.

Definition 2.5.1. *Duty Tree: A duty tree is a directed out-tree $T = (N, A)$ with the root node s representing a flight f, other nodes $i, i \in N \backslash \{s\}$ representing flights that can be connected from f, and arcs $A = (i, j), i \neq j, i \in N, j \in N$ representing legal day connections from flight i to flight j. The unique path from node s to every other node $i, i \in N \backslash \{s\}$ is a directed path that represents a legal duty starting from flight f.*

To build the duty trees, the concept of a flight's day connections is often used. Given the set of flights, the day connection list is based on the departure and arrival stations, and the sit time limitations. For example, based on minimum and maximum sit times, usually defined in an airline's rules, we can obtain any flight's connection flight list that meets minimum and maximum sit time requirements. As illustrated in Figure 2.6, flight 1 is an arriving flight, while flights 2, 3, 4, 5 and 6 are departing flights. The connection time between the arriving flight and each departing flight is different. Based on the connection time rule for minimum sit time and maximum sit time, only flights 3, 4, and 5 can be legally connected to flight 1, so the day connections for flight 1 are flights 3, 4 and 5.

For each flight $f, f \in F$, we can build a duty tree as in Algorithm 2.1. Given a set of flights, suppose the corresponding day connection flight list for each flight has been built, and the maximum number of flights allowed in a duty is also known

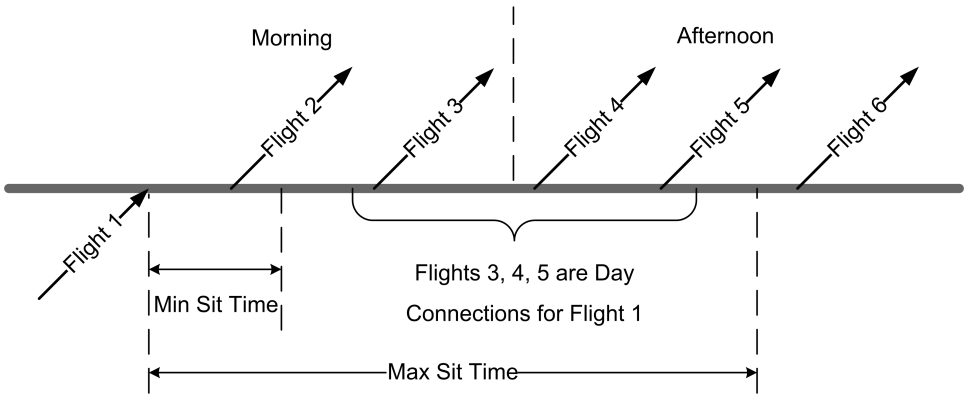

Figure 2.6: A flight's day connections.

in advance. Then, we can build the duty tree for each flight through a recursive enumeration process while checking the duty legality for each candidate.

For 10 flights $f_1, f_2, ..., f_{10}$ that can be connected in a certain way based on day connections, after the duty related legality check, resulting duties with flights connected are shown in Table 2.1. Figure 2.7 shows a duty tree for root flight f_1.

Table 2.1: The duties starting from flight f_1.

Index	Duty List
1	f_1
2	f_1, f_2
3	f_1, f_2, f_3
4	f_1, f_2, f_3, f_4
5	f_1, f_2, f_3, f_5
6	f_1, f_6
7	f_1, f_6, f_7
8	f_1, f_6, f_8
9	f_1, f_6, f_8, f_9
10	f_1, f_6, f_8, f_{10}

The duty tree has the following properties:

Property 2.5.1. *Duty Tree Properties:*

a) Each flight f_i has its own duty tree, in which it serves as the root node, and the

Algorithm 2.1 Build a Duty Tree

Require: The set of flights F, the day connection flight list DC_i for each flight $f_i, f_i \in F$, the maximum number of flights allowed in a duty K.

```
for f_i ∈ F do
    CURRENT: = f_i
    save CURRENT into THREAD
    Let DC_i be the day connections of flight f_i
    if DC_i ≠ ∅ then
        for f_j ∈ DC_i do
            enumeration(CURRENT, f_j)
        end for
    end if
end for
The legal duties for flight f_i are all represented in the duty tree T_{f_i}, which is stored in THREAD.
..... subroutine .....
enumeration(CURRENT, f_j)
add f_j to CURRENT
check duty legality on CURRENT
if legal then
    save CURRENT into THREAD
    if length(CURRENT) < K then
        Let DC_j be the day connections of flight f_j
        if DC_j ≠ ∅ then
            for f_k ∈ DC_j do
                enumeration(CURRENT, f_k)
            end for
        end if
    end if
end if
remove f_j from CURRENT
```

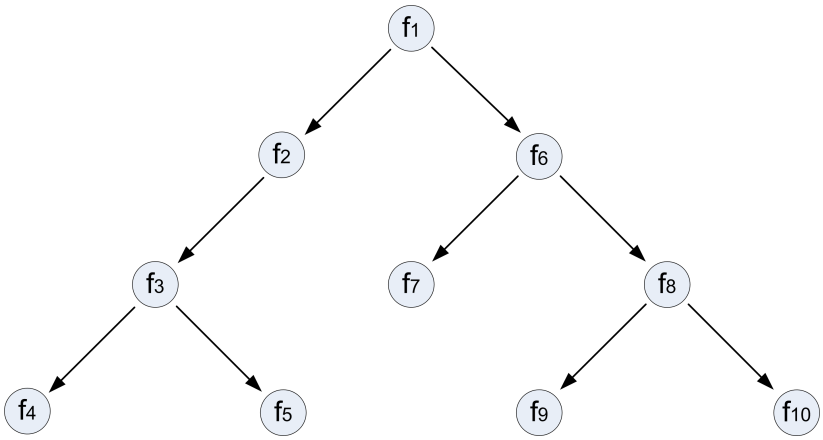

Figure 2.7: The duty tree rooted at flight f_1.

path from the root to any node in depth-first order in the duty tree represents a legal duty.

b) Each node corresponds to a flight, and a flight f can have multiple nodes in the duty tree. If node i corresponds to flight f, there will not be a parent/ancestor or child/descendant node corresponding to the same f.

c) The height of a duty tree depends on the maximum number of flights allowed in a duty as defined by airline rules.

d) The number of duties represented by a duty tree is equal to the number of nodes in the duty tree, and the set of all duties is the union of duty sets from all duty trees.

e) Completeness: All legal duties starting from flight f are represented in the duty tree for flight f.

f) Exclusiveness: Each duty will appear in one and only one duty tree, that is, the duty tree for the first flight in the duty.

g) Tail off: If a duty $d = \{f_1, ..., f_j, f_k, f_m\}$ is legal and the tail node f_m is removed from the duty, the remaining nodes $\{f_1, ..., f_j, f_k\}$ still comprise a legal duty; if f_k is removed, $\{f_1, ..., f_j\}$ is still a legal duty, and so on.

After building the duty tree structure, all legal duties are represented in the tree and can be listed explicitly using depth-first search.

2.5.3 Speed up using End Duty Tree

Because of the particular nature of the crew pairing problems, each pairing must start at a crew base and usually must end at the same crew base. In this case, either the flight's departure station is the crew base and the whole tree will be included when building a pairing's first duty, or the flight's departure station is different from the crew base, in which case, the whole tree will not be included when building a pairing's first duty. For the last duty in a pairing, the case can be different because the last duty can start at any station other than the crew base, so any flight starting from a non-base station can become the first flight in the last duty. While it should end at the crew base, in this case, the duties represented by the last duty tree may be a subset of the duties represented by a flight's whole tree. The concept of the end duty tree is defined in Definition 2.5.2.

Definition 2.5.2. *End Duty Tree: A subtree of a flight's duty tree with leaf nodes consisting of only flights that arrive at the crew base cb.*

For each flight $f_i, f_i \in F$, we have an end duty tree for each crew base and we can build the tree as in Algorithm 2.2:

Algorithm 2.2 Build the End Duty Tree

Require: The set of flights F, each flight's duty tree T_f, crew base *cb*.

Start from the corresponding duty tree T_f.

repeat

Check all the leaf nodes, delete those leaf nodes with arrival airport being different from the crew base *cb*.

until all leaf nodes have arrival airport being the crew base *cb*.

All legal end duties corresponding to crew base *cb* are represented in this end duty tree.

Figure 2.8 shows the basic structure of the end duty tree. That is, all the leaf nodes end at the crew base while the other nodes may or may not end at the crew base.

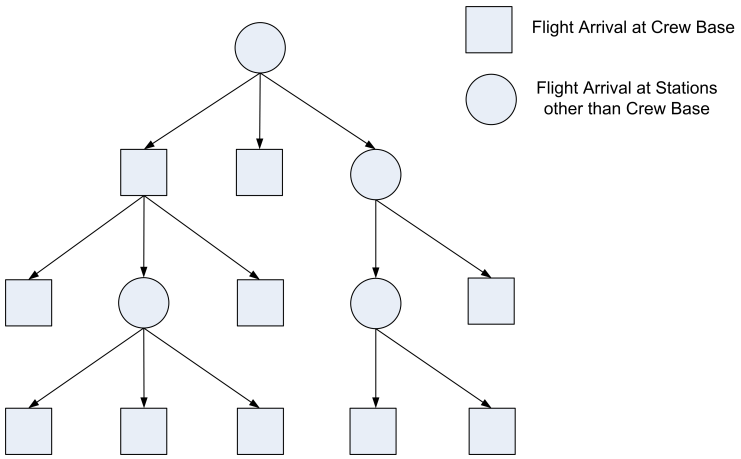

Figure 2.8: The end duty tree.

The end duty tree has the following properties:

Property 2.5.2. *End Duty Tree Properties:*

a) For each duty tree, there is one and only one corresponding end duty tree for each crew base.

b) The leaf nodes correspond to flights that arrive at the crew base while other nodes other than the leaf nodes correspond to flights that arrive at any other airport.

c) The flight corresponding to the parent of a leaf node will not end at a crew base.

d) There is no tail off effect for the end duty tree. If the last node is removed from the nodes corresponding to a duty, the remaining flights will not be an end duty. Nevertheless, if we remove more than two or more nodes from an end duty, the remaining nodes may constitute another end duty.

The relationship between a flight's duty tree and the end duty tree is illustrated in Figure 2.9. For each flight f, we have one duty tree, but we may have multiple corresponding end duty trees if we have multiple crew bases for the pairing problem to be solved, with each end duty tree associated with each crew base.

For example, we have nine flights indexed as 9, 10, 11, 12, 19, 28, 34, 55, and 56. Figure 2.10 shows a duty tree rooted at flight 9, with detailed departure and arrival

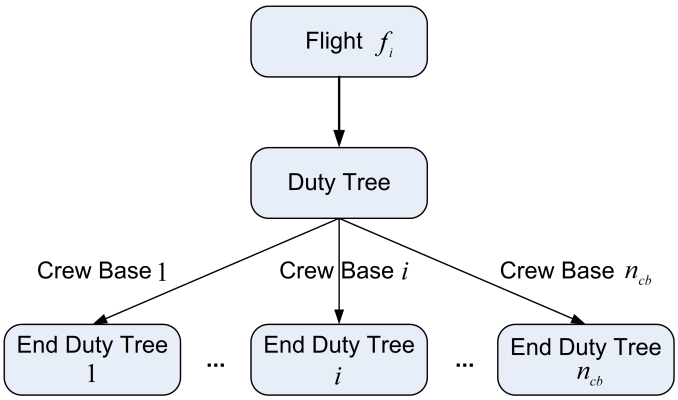

Figure 2.9: The relationship between duty tree and end duty tree.

times and stations. The corresponding duties contained in the duty tree are listed in Table 2.2. Figure 2.11 shows the duty tree and the resulting end duty tree for Atlanta crew base ATL. There are only 3 nodes in the end duty tree that corresponds to one end duty {9, 10, 55 } ending at the ATL airport.

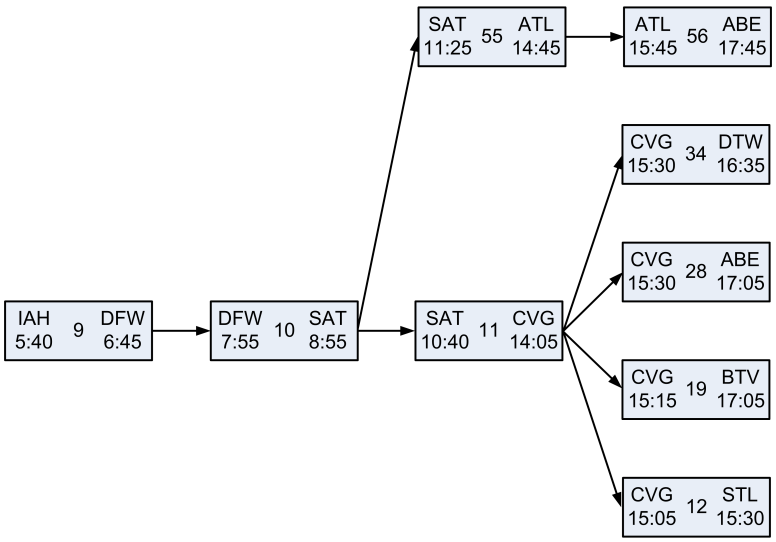

Figure 2.10: A duty tree with nine flights.

2.5.4 Crew Pairing Legality

Crew pairing problems are characterized with having many rules that define whether a pairing is legal, that is, whether the pairing can be operated by a crew. There are

Table 2.2: The duties represented by the duty tree.

Duty Index	Flight Index	Dept Sta	Dept Time	Arrv Sta	Arrv Time
duty 1	9	IAH	0540	DFW	0645
duty 2	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
duty 3	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	11	SAT	1040	CVG	1405
duty 4	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	11	SAT	1040	CVG	1405
	12	CVG	1505	STL	1530
duty 5	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	11	SAT	1040	CVG	1405
	19	CVG	1515	BTV	1705
duty 6	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	11	SAT	1040	CVG	1405
	28	CVG	1530	ABE	1705
duty 7	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	11	SAT	1040	CVG	1405
	34	CVG	1530	DTW	1635
duty 8	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	55	SAT	1125	ATL	1445
duty 9	9	IAH	0540	DFW	0645
	10	DFW	0755	SAT	0855
	55	SAT	1125	ATL	1445
	56	ATL	1545	ABE	1745

duty period level rules and pairing level rules. For example, some basic restrictions on the duty periods include minimum and maximum sit times between connecting flight legs, and maximum amount of flying time in a duty period being limited for safe operating purposes. For example, when pairings are built from duties, the following pairing level legality rules could apply:

- Minimum and maximum overnight rest between duties in a pairing.
- Maximum number of duties in a pairing.
- Maximum pairing length in calendar days.
- Maximum duty period time in a pairing.

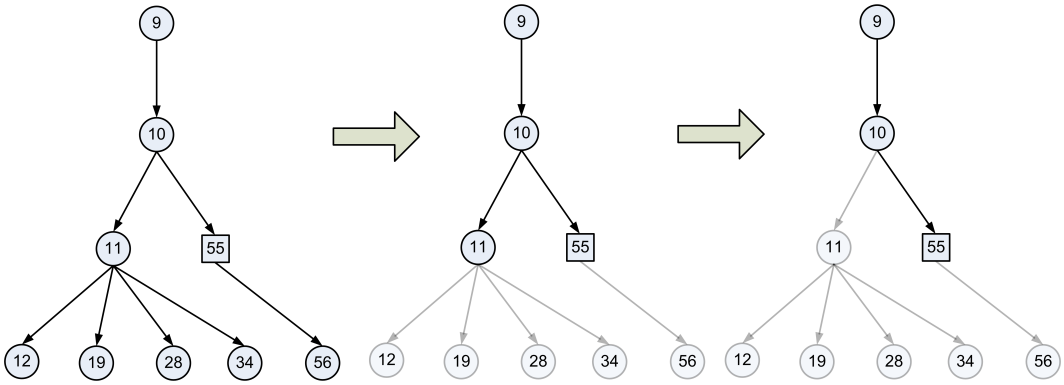

Figure 2.11: The duty tree and corresponding end duty tree.

- The 8-in-24 rule, which requires a longer overnight rest time (compensatory rest) if there are more than 8 hours of flying time in any 24-hour window.
- Many more from government, labor union, and airline operational rules.

Duty level rules are checked when the duty trees T associated with each flight leg f is built. The duties obtained from the duty trees are all legal duties. Next, we build pairings based on legal duties from these duty trees. Suppose we have $d_1 \in T_1$, $d_{2a} \in T_2$, and $d_{2b} \in T_2$, and they are all legal duties. However, d_1, d_{2a} can form a legal pairing while d_1, d_{2b} will result in an illegal pairing because it may violate the maximum duty period time allowed in a pairing. Though d_{2a} and d_{2b} come from the same duty tree T_2 and both are legal, they will have different roles when being connected with $d_1 \in T_1$. We will use a subtree concept to address this situation resulted from the legality check in the duty tree approach.

Some of the crew pairing rules will be illustrated below using FAA new rules. In 2011, FAA announced a new rule that revises commercial passenger airline pilot scheduling to ensure pilots have a longer time for rest before they enter the cockpit. This new rule raises the safety bar to prevent fatigue. Key components of the new rule for commercial passenger flights include the following [1].

- Varying flight and duty requirements based on the time the pilot's day begins:

The new rule sets different requirements for pilot flight time, duty period and rest based on the time of day pilots begin their first flight, the number of scheduled flight segments and the number of time zones they cross.

- Flight duty period: The allowable length of a flight duty period depends on when the pilot's day begins and the number of flight segments expected.

- Flight time limits of eight or nine hours: The flight time, including when the plane is moving under its own power before, during, or after a flight, is limited to eight or nine hours depending on the start time of the pilot's entire flight duty period.

- 10-hour minimum rest period: The rule sets a 10-hour minimum rest period prior to the flight duty period, a 2-hour increase over the old rules.

- New cumulative flight duty and flight time limits: To address potential cumulative fatigue, the new rule places weekly and 28-day limits on the amount of time a pilot may be assigned any type of flight duty; 28-day and annual limits on actual flight time; weekly minimum consecutive free hours, and so on.

For unaugmented operations, some detailed maximum flight time limits are listed in Table 2.3, and the maximum flight duty period limits are provided in Table 2.4. These restrictions are an attempt to address flight crew members' circadian rhythms, fatigue caused by longer amount of time spent at work, and the number of takeoffs and landings that are task intensive and safety critical.

The extended flight duty periods accommodate common operational practices for which the carrier provides additional crew and adequate on-board rest facilities, such as a class 1 facility with a bunk or other surface that allows for a flat sleeping position and is separate from both the flight deck and the passenger cabin, a class 2 facility with a seat in an aircraft cabin that allows for a flat or near flat sleeping position,

and a class 3 facility with a seat in an aircraft cabin or flight deck that reclines at least 40 degrees and provides leg and foot support [1]. Some detailed maximum flight duty period limits for augmented operations are given in Table 2.5.

The new rule will take effect in two years to allow commercial passenger airline operators time to transition.

Table 2.3: Maximum flight time limits for unaugmented operations.

Time of Report (Acclimated)	Maximum Flight Time (hours)
0000-0459	8
0500-1959	9
2000-2359	8

Table 2.4: Maximum flight duty period limits for unaugmented operations.

Scheduled Time of Start (Acclimated Time)	Maximum Flight Duty Period (hours) For Lineholders based on Number of Flight Segments						
	1 seg	2 segs	3 segs	4 segs	5 segs	6 segs	7+ segs
0000-0359	9	9	9	9	9	9	9
0400-0459	10	10	10	10	9	9	9
0500-0559	12	12	12	12	11.5	11	10.5
0600-0659	13	13	12	12	11.5	11	10.5
0700-1159	14	14	13	13	12.5	12	11.5
1200-1259	13	13	13	13	12.5	12	11.5
1300-1659	12	12	12	12	11.5	11	10.5
1700-2159	12	12	11	11	10	9	9
2200-2259	11	11	10	10	9	9	9
2300-2359	10	10	10	9	9	9	9

The duty tree approach proposed in this study has the advantage of being flexible to incorporate any kind of rules.

2.5.5 Sub Duty Tree and Sub End Duty Tree

Because of the pairing level legality check, sometimes a partial duty tree is valid for use in the construction of legal pairings. A subtree concept is used to represent this

Table 2.5: Maximum flight duty period limits for augmented operations.

Scheduled Time of Start (Acclimated Time)	Maximum Flight Duty Period (hours) based on Rest Facility Class (C) and Number of Pilots (P)					
	C1 & 3P	C1 & 4P	C2 & 3P	C2 & 4P	C3 & 3P	C3 & 4P
0000-0559	15	17	14	15.5	13	13.5
0600-0659	16	18.5	15	16.5	14	14.5
0700-1259	17	19	16.5	18	15	15.5
1300-1659	16	18.5	15	16.5	14	14.5
1700-2359	15	17	14	15.5	13	13.5

situation. From [5], we have the basic definition of subtree:

Definition 2.5.3. *Subtree: A connected subgraph of a tree is a subtree [5].*

As noted, each duty tree rooted at flight $f, f \in F$ is represented as T_f. In this case, ST_f is used for the subtree of T_f, and each subtree is represented by a mask.

Definition 2.5.4. *Mask: A mask is an array indicating the number of duties to be taken and the number of duties to be skipped from the whole duty list corresponding to T_f. It takes the format as:*

$$mask = [\ num\ taken,\ \ num\ skip,\ \ num\ taken,\ \ num\ skip,\ .....]$$

The subtree has the following properties:

Property 2.5.3. *Subtree Properties:*

a) It is resulted from the legality check, and is applicable to both the duty tree and the end duty tree.

b) It is represented as a mask with a keep, skip scheme.

c) The sum of the number taken and number skipped is equal to the total number of duties in that corresponding duty tree.

For example, in a pairing, suppose we already have the first duty $\hat{d}$ whose last flight is f_0, one of the night connections of f_0 is f_1, and the duty tree starting from f_1 is as shown in Figure 2.7. While the duty tree contains all legal duties, let the represented duty set be $D = \{d_1, ..., d_n\}$. Nevertheless, the partial pairing $[\hat{d}, d_i], i \in \{1, ..., n\}$ may not be entirely legal according to pairing legality rules. Among the set $D = \{d_1, ..., d_n\}$, suppose there is a subset of D constituting a legal partial pairing when being connected with $\hat{d}$. Then we use a mask to specify which one is legal for the pairing construction. After the legality check, suppose we get two subtrees as in Figure 2.12. The corresponding duties represented in the two subtrees are as shown in Table 2.6.

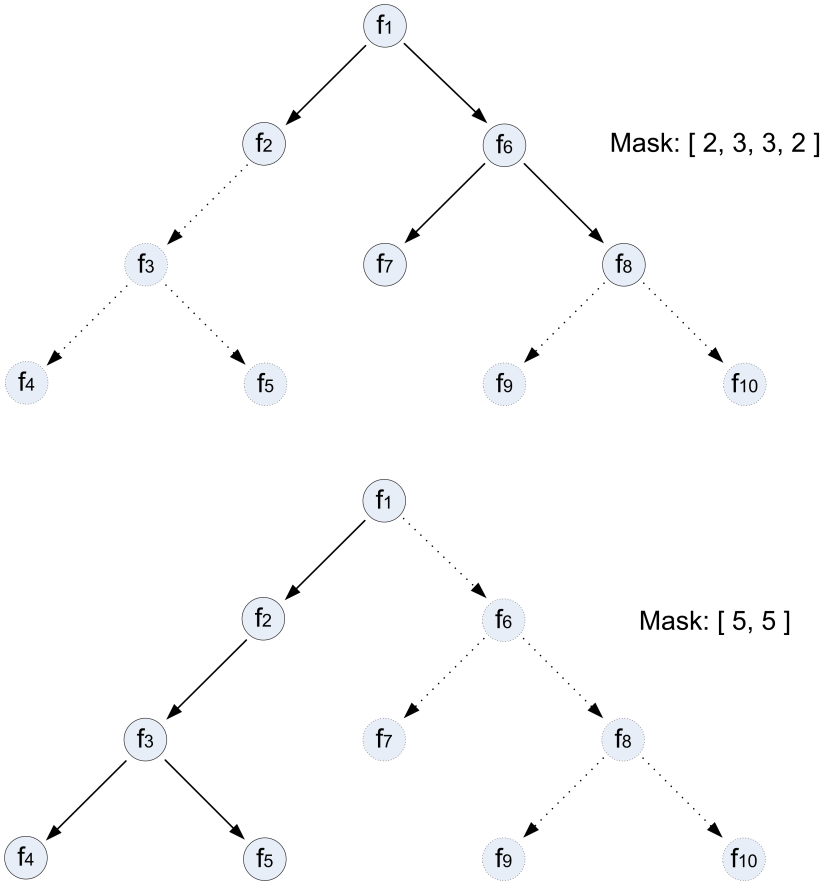

Figure 2.12: Subtree mask examples.

Subtree mask [2 , 3 , 3 , 2] indicates taking the first two duties, skipping next

Table 2.6: Subtree mask examples.

Duties	Legality Case 1	Legality Case 2
f_1	legal	legal
f_1, f_2	legal	legal
f_1, f_2, f_3	illegal	legal
f_1, f_2, f_3, f_4	illegal	legal
f_1, f_2, f_3, f_5	illegal	legal
f_1, f_6	legal	illegal
f_1, f_6, f_7	legal	illegal
f_1, f_6, f_8	legal	illegal
f_1, f_6, f_8, f_9	illegal	illegal
f_1, f_6, f_8, f_{10}	illegal	illegal
Subtree mask	[2, 3, 3, 2]	[5, 5]

three duties, taking the following three duties, and skipping the last two duties in the whole duty list represented in f_1's duty tree.

Subtree mask [5 , 5] indicates taking the first five duties, and skipping the last five duties from the whole duty list in f_1's duty tree.

2.5.6 Partial Pairings

A crew pairing consists of one or several duties, with the first duty starting from the crew base, the last duty ending at the same crew base, and other duties starting or ending at airports other than the crew base. These other airports are usually called the layover cities. To build pairings from duty trees, we need the concepts of night connections and partial pairing.

Given the set of flights, we build the night connection list based on the departure and arrival stations and the layover time limitations. Using minimum and maximum layover times, we can get the night connections for a flight. In Figure 2.13, the night connections for flight 1 are flights 4, 5 and 6 while flight 6 is a double overnight case.

Partial pairing: A partial pairing starts from a crew base *cb*, with one or more duties, and ends at an airport other than the crew base *cb*. That is, the last duty in

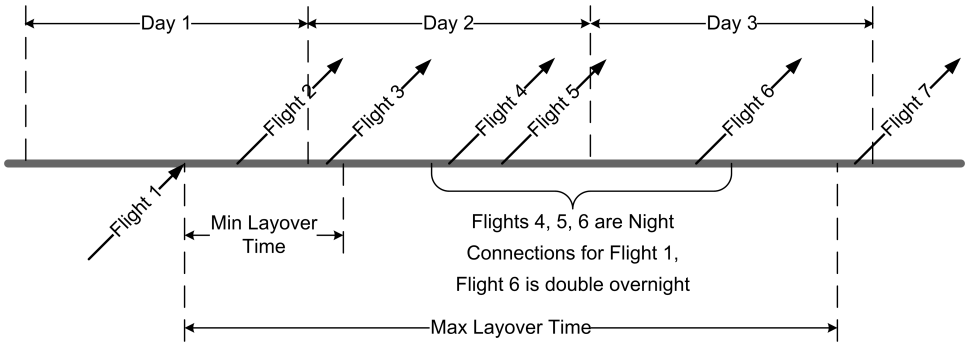

Figure 2.13: A flight's night connections.

a partial pairing ends at an airport other than the crew base.

k-duty partial pairing: This term refers to a partial pairing containing k duties, where k is less than the maximum number of duties K_d allowed in a legal pairing. One-duty partial pairing is always legal. Because a one-duty partial pairing has only one duty, the duty is legal, so the partial pairing is also legal.

The partial pairing used here indicates that the list of duties built so far are legal in terms of pairing rules, and they can be extended to a pairing that may be legal or illegal when one or more duties are added. The partial pairing enables us to describe different cases in the process of pairing construction.

(i) Construct one-duty partial pairing

Given a crew base cb, let F_{cb} be the set of flights departing from the crew base. For each flight $f \in F_{cb}$, let T_f be the corresponding duty tree for flight f. Traverse duty tree T_f using depth first search and get the list of duties D_{T_f} contained in T_f. Let set $D_{T_f,cb}$ be the set of duties in D_{T_f} and end at crew base cb, and $D_{T_f,\overline{cb}}$ be the set of duties in D_{T_f} and not end at crew base cb. We have $D_{T_f,\overline{cb}} \subseteq D_{T_f}$, $D_{T_f,cb} \subseteq D_{T_f}$, $D_{T_f,\overline{cb}} \cup D_{T_f,cb} = D_{T_f}$, $D_{T_f,\overline{cb}} \cap D_{T_f,cb} = \emptyset$. If $D_{T_f,\overline{cb}} = \emptyset$, then $D_{T_f,cb} = D_{T_f}$.

For each duty $d \in D_{T_f,cb}$, it starts and ends at the same crew base and becomes a one-duty legal pairing that will not be extended. For each duty $d \in D_{T_f,\overline{cb}}$, we

create a one-duty partial pairing $p = \{d\}$.

(ii) Construct the k-duty partial pairing for $1 < k < K_d$, where K_d is the maximum number of duties allowed in a legal pairing.

Given a (k-1) duty partial pairing $p = \{d_1, ..., d_{k-1}\}$, let the night connection be $NC_{\tilde{f}}$ for p, where $\tilde{f}$ is the last flight in partial pairing p.

For each flight $\hat{f} \in NC_{\tilde{f}}$, let $T_{\hat{f}}$ be the duty tree for flight $\hat{f}$. We iterate through duties in duty tree $T_{\hat{f}}$ using a depth-first search. For each $\hat{d} \in T_{\hat{f}}$, we extend p to construct a k duty pairing or partial pairing $\hat{p} = \{d_1, ..., d_{k-1}, \hat{d}\}$, check its legality, and store the result using the mask scheme, that is, number taken, number skipped. Then we have the following cases:

(a) If $\hat{p}$ is illegal in terms of pairing legality, then duty $\hat{d}$ is considered to be skipped;

(b) If $\hat{p}$ is legal and ends at a station other than the crew base, then $\hat{p}$ is a legal partial pairing, and $\hat{d}$ is considered to be taken;

(c) If $\hat{p}$ is legal and ends at the crew base, then $\hat{p}$ is a legal pairing, and $\hat{d}$ is considered to be taken.

A subtree $ST_{\hat{f}}$ is associated with a partial pairing $p = \{d_1, d_2, ..., d_k\}$. Duties contained in a subtree are a subset of all duties contained in the corresponding duty tree $T_{\hat{f}}$. For those duties in the subtree, each can extend the partial pairing p, such that $\hat{p} = \{d_1, d_2, ..., d_k, \hat{d}\}, \hat{d} \in ST_{\hat{f}}$ is a legal pairing or partial pairing.

The duties $\hat{d} \in T_{\hat{f}} \backslash ST_{\hat{f}}$ are skipped because they cannot extend this partial pairing p, that is, $\{d_1, d_2, ..., d_k, \hat{d}\}, \hat{d} \in T_{\hat{f}} \backslash ST_{\hat{f}}$ is not a legal pairing or partial pairing.

2.5.7 The Pairing Tree

All pairings belonging to a crew base *cb* are stored in a pairing tree, rooted at the crew base *cb*, as shown in Figure 2.14. In the figure, T_i is used to represent duty

trees, ST_i to represent sub duty trees and $\overline{ST_i}$ to represent sub end duty trees. As the figure shows, the pairing tree has the basic building blocks of the duty tree or sub duty tree or sub end duty tree, as discussed in previous sections.

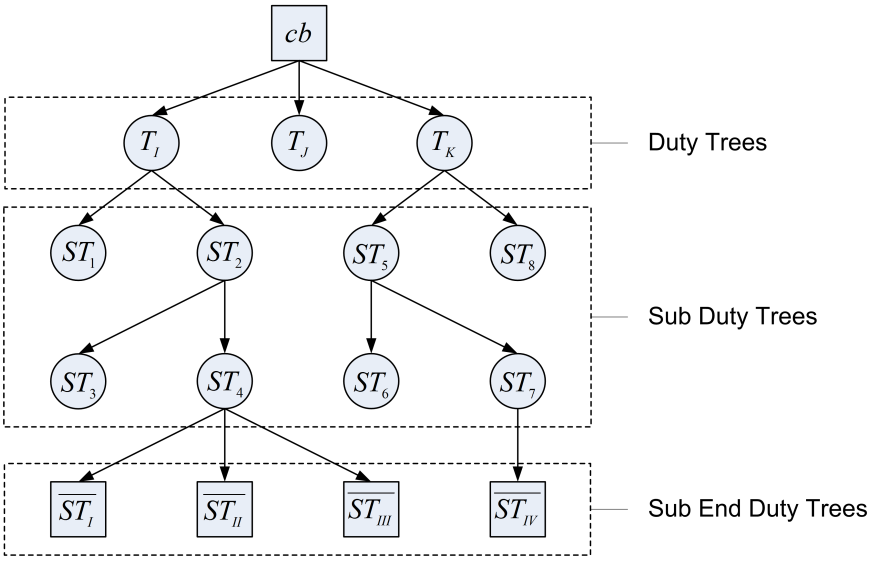

Figure 2.14: A four-duty pairing tree.

The pairing tree has the following properties:

Property 2.5.4. *Pairing Tree Properties:*

a) Each pairing tree is rooted at its crew base.

b) The maximum depth of a pairing tree is the maximum number of duties K_d in a pairing according to airline rules.

c) Nodes in a pairing tree are duty trees, sub duty trees or sub end duty trees.

d) Nodes in depth 1 are all duty trees.

e) Nodes in depth K_d are all sub end duty trees.

f) Nodes on depth 2 and $K_d - 1$ as well as in between are all sub duty trees.

Considering that the number of partial pairings are large, especially for k-duty partial pairings when $k > 1$, we use a compact storage method to store masks for each

$k \in 2, ..., K_d$ using two one-dimensional arrays, that is, $BegMask$ and $IndMask$.

Let $nMask$ denote the number of masks for all partial pairings. The array $BegMask$ is of length $nMask$ and holds all masks' beginning indices in a left to right order. The array $IndMask$ is of length $m + 1$, where m is the sum of the length of all masks. $BegMask[i]$ contains the beginning index of the i^{th} mask, and $BegMask[i + 1] - 1$ contains the end index of the i^{th} mask. $IndMask[BegMask[i]]$ to $IndMask[BegMask[i + 1] - 1]$ contains the details of the i^{th} mask.

Let the set of partial pairings be P and $|NC_p|, p \in P$ be the number of night connections for partial pairing p. We have $nMask = \sum_{p \in P} |NC_p|$.

We traverse the pairing tree using the depth-first search. The first mask is the subtree mask for the first night connection for the first partial pairing, the second mask is the subtree mask for the second night connection for second partial pairing, and so on. The mask storage scheme is illustrated in Figure 2.15. For example, for three partial pairings, the first has three night connections, and the BegIndex for the first night connection indicates a subtree mask detailed with [5, 4, 2, 4]. The BegIndex for the second night connection points to a subtree mask detailed with [7, 1, 2, 5], and so on.

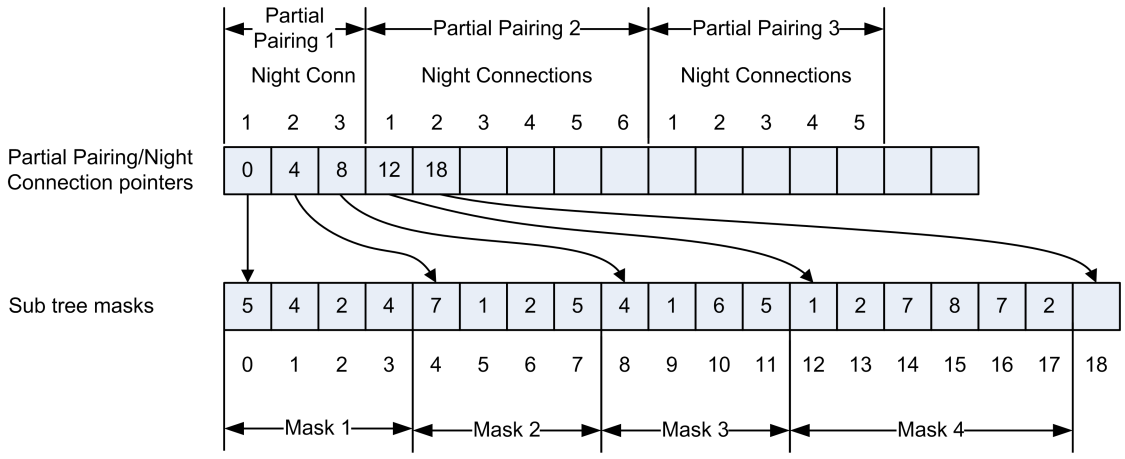

Figure 2.15: The mask storage.

2.5.8 Computational Results

We conducted some experiments on the pairing enumeration based on duty trees. The machine used is an Intel Core2 Quad CPU Q8300 @ 2.50GHz, RAM 8.00GB, Windows7 Home Premium 64-bit Operating System. Some computational results including the total number of duties, pairings, last duties, and masks generated, and the computer memory used to build the duty trees for a couple of daily crew pairing problems are provided in Table 2.7 and Table 2.8.

Table 2.7: Results for building duty trees for 350-flight and 212-flight problems.

Features	350-flight problem	212-flight problem
Number of flights	350	212
Number of crew bases	3	4
Total number of duties generated	9,341	6489
Total number of pairings generated	3,303,013	5,052,622
Total number of last duties generated	1572	1290
Total number of masks generated	5503	11827
Computer memory used (less than)	20 MB	25 MB

Table 2.8: Results for building duty trees for 219-flight problem.

Features	219-flight problem 1	219-flight problem 2
Number of flights	219	219
Number of crew bases	4	4
Total number of duties generated	11,425	21,294
Total number of pairings generated	75,982,683	657,531,006
Total number of last duties generated	2189	4,012
Total number of masks generated	64988	181,646
Computer memory used (less than)	55 MB	200 MB

The 350-flight problem has three crew bases and 350 flights. The total number of duties generated is 9341 and the total number of masks generated is 5503. The number of pairings is 3.3 million and the computer memory used to build the duty

trees is less than 20 MB. For the 212-flight problem run with four crew bases and 212 flights, the duty tree contains 5 million pairings, and the computer memory used is less than 25 MB.

We also tried another problem with 219 flights and four crew bases and setting more relaxed day connection and night connection rules. In the first run, we build the duty tree for 11,425 duties and above 75 million pairings, the computer memory used is 55 MB. For the second run, we can see that the computer memory used to handle the duty tree is still very manageable. Even with 21,294 duties and 657,531,006 pairings, it is still less than 200 MB.

2.5.9 Summary

In this research, we propose a new duty tree approach that can potentially reduce the high memory storage requirement when constructing the duties and pairings and conducting the subsequent optimization process. That is, in the proposed method, a huge number of pairings, potentially billions, can be represented in the duty tree structure and can be handled in the computer memory, as shown in the computational results. This approach enables solving bigger crew pairing problems without explicit storage of the pairings. In addition, a legality check has been conducted beforehand for all duties and pairings in the duty tree scheme, so the pairings from the pairing tree constructed upon the duty trees are all legal pairings. The significant savings in memory can potentially contribute to approaches for solving bigger crew pairing optimization problems.

CHAPTER III

CREW PAIRING OPTIMIZATION USING PRIMAL-DUAL SUBPROBLEM SIMPLEX METHOD BASED ON DUTY TREE

3.1 Introduction

The set partitioning formulation provides a powerful and popular approach to solving crew pairing optimization problems because it enables handling complex crew-related rules during the pairing construction and a huge number of columns during the optimization process. The details are discussed in the following section.

3.2 Crew Pairing Optimization with Set Partitioning Formulation

The crew pairing optimization problem is usually formulated as a set partitioning problem as follows. The objective is to find a set of good pairings that minimize the pairing costs while ensuring each flight segment is covered by the number of crews required.

$$\begin{aligned} \text{Min} \quad & \sum_j c_j x_j \\ \text{s.t.} \quad & \sum_j a_{ij} x_j = b_i, \forall \text{ flight segment } i \qquad (3.1) \\ & x_j \in \mathbb{B}, \forall \text{ pairing } j \qquad (3.2) \end{aligned}$$

where:

c_j: the cost of pairing j.

b_i: the number of crews required on flight segment i.

$$a_{ij} = \begin{cases} 1, & \text{if pairing } j \text{ covers flight segment } i, \\ 0, & \text{otherwise .} \end{cases}$$

x_j: decision variable, $x_j = \begin{cases} 1, & \text{if pairing } j \text{ is selected ,} \\ 0, & \text{otherwise .} \end{cases}$

This formulation has a coefficient matrix with a row for each flight segment and a column for each pairing. The entries in the matrix are 0 or 1, with a 1 signifying that the pairing corresponding to the column includes the flight segment corresponding to the row.

The TRIP Approach

To solve crew pairing problems, a trip reevaluation and improvement program (TRIP) was developed and applied at American Airlines [7]. The TRIP approach starts with an initial feasible solution and tries to improve the solution using a local optimization approach, that is, a subproblem that corresponds to a subset of pairings is selected, pairings are generated based on the flight segments in the selected subproblem, and the set-partitioning formulation is used to obtain a set of pairings that minimizes the subproblem costs and cover the selected flight segments. If an improved solution of the subproblem is found, then the new pairings obtained from the current subproblem solution, together with other pairings not involved in the current subproblem, form a new set of pairings for the entire problem. The selection, generation, and solving procedures are repeated until certain stopping criteria are reached.

The TRIP approach was reported to achieve big savings for American Airlines in terms of crew resource utilization [7]. Nevertheless, the TRIP approach may lead to suboptimum solutions with local minima because of the limited effective subproblem size.

A Global Optimization Approach with Primal Subproblem Simplex Method

To overcome the limitation of the TRIP approach introduced above, Anbil et al.

[8] proposed a global optimization approach, in which a primal subproblem simplex method that takes the entire problem into consideration is discussed. After generating millions of pairings that correspond to millions of columns in the entire problem, this method takes advantage of the SPRINT approach of John Forrest to solve the linear programming (LP) relaxation of the set-partitioning formulation, that is, solving a subproblem consisting of a small subset of the columns by primal subproblem simplex method and using the optimal dual vector from subproblems to price out all columns of the entire problem. The columns for the next subproblem are selected from those with the smallest reduced costs. The procedures are repeated until an optimal LP solution for the entire problem is reached, and the integer solution is obtained using a follow-on fixing idea.

In this method, the optimal dual solution from a subproblem is used to price out columns to construct the next subproblem with negative reduced costs. The concern here is that this dual solution is somewhat local in terms of the entire problem because it is obtained when many other dual constraints corresponding to primal columns not involved in the subproblem are relaxed. Before the optimal solution of the entire problem is reached, the subproblem dual solution from iteration to iteration is infeasible in terms of the entire problem.

Primal-Dual Subproblem Simplex Method

Hu [64], Hu and Johnson [65] proposed a primal-dual subproblem simplex method to solve general linear programs with large number of columns. They applied the method to crew pairing optimization. This method starts from a dual of the feasible solution. The subproblem is constructed from those columns with the smallest reduced costs and solved by a LP solver - IBM Optimization Subroutine Library. The optimal dual vector from the subproblem is used to improve the global feasible dual solution of the entire problem, which is then used to price out all the columns of the entire problem. The procedures are repeated from iteration to iteration until an

optimal solution is achieved.

For the large number of pairings generated, the explicit storage of each pairing requires large storage space and a long time for the pricing out. To solve such a large pairing problem efficiently, Hu and Johnson [65] developed a compact storage scheme for the crew pairing matrix, and solved the entire crew pairing problem by combining the primal-dual subproblem simplex method and the compact storage scheme. In the approach, the pairing set is constructed first, then the primal-dual subproblem simplex method together with a compact storage scheme is used to solve to optimality. Many computational results are also provided in details in the work.

Primal-Dual Subproblem Simplex Method with Column Generation

With the goal of solving larger problems within reasonable amounts of time, Shaw [104] proposed a method to perform delayed column generation within the primal-dual subproblem simplex method so that a complete enumeration of all possible columns is not required. The duty network was proposed with an enhanced compact storage scheme based on a string network. The techniques for using shortest path algorithms to construct subproblems, improve the dual feasible solution, and obtain the next iteration subproblem in the primal-dual subproblem simplex method were described, and the application for airline crew scheduling was discussed.

3.3 Primal-Dual Subproblem Simplex Method for Crew Pairing Optimization

The simplex method was pioneered by Dantzig in 1947 and became one of the most widely used solution methods for linear programs. The method uses a sequence of pivot operations to improve the objective value of the targeted problem. The primal-dual algorithm was originated with Dantzig et al. [40] and has wide application.

Consider the LP in its standard form:

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where the cost vector $c^T \in \mathbb{R}^n$, the constraint coefficient matrix $A \in \mathbb{R}^{m \times n}$, the right-hand side vector $b \in \mathbb{R}^m$, and the decision vector $x \in \mathbb{R}^n$.

Let's call the above LP as a primal problem (**P**). From the duality theory, the dual of (**P**) is to find a vector π by solving the following problem:

$$\begin{aligned} \max \quad & \pi b \\ \text{s.t.} \quad & \pi A \leq c \end{aligned}$$

where $\pi^T \in \mathbb{R}^m$. Let's denote the dual problem as (**D**).

For the above primal (**P**) and its dual (**D**), we have the following complementary slackness conditions:

Theorem 3.3.1. *Complementary Slackness: A pair x, π, when x is feasible to the primal* (**P**) *and π is feasible to the dual* (**D**)*, is optimal if and only if*

$$\pi_i(a_i x - b_i) = 0, \ \forall \ i$$
$$(c_j - \pi A_j)x_j = 0, \ \forall \ j$$

The primal-dual simplex method [90] is to search for a feasible x, starting from a vector π_0 that is feasible to the dual (**D**), that is, $\pi_0 A_j \leq c_j, j \in \{1, 2, ..., n\}$, where A_j denotes the j^{th} column of the matrix A, by solving a problem called the *restricted master problem*, as follows,

$$
\begin{aligned}
\min \quad & \xi = \sum_{i=1}^{m} y_i \\
\text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j + y_i = b_i, i = 1, ..., m \\
& x_j \geq 0, j \in J \\
& x_j = 0, j \notin J \\
& y_i \geq 0.
\end{aligned}
$$

where $J = \{j : \pi^{'} A_j = c_j\}$.

The restricted master problem consists of a subset of columns satisfying $c_j - \pi A_j = 0$ from the original problem. If we can find an optimal x for the restricted master problem such that $\xi_{opt} = 0$, then x and π is a pair of the optimal solution for the primal and dual problems. Otherwise, we can obtain information from the dual of the restricted master problem and use it to update π. The procedure of solving the restricted master problem, improving the dual, and redefining columns is repeated until a certain stopping criterion, such as optimality or infeasibility, is reached. The overall method procedures are described in Algorithm 3.1.

The primal-dual algorithm correctly solves (**P**) in a finite amount of time [90]. The primal-dual algorithm enables solving an LP problem by solving a feasibility problem.

Hu [64], Hu and Johnson [65] developed a primal-dual subproblem simplex method to solve linear programming problems with few rows and many columns. In this method, the restricted master problem is constructed using the columns with reduced costs within a small cut-off, including zeros. More columns are included in the subproblem at each iteration, and the method converges faster in general. In this present work, we are using the primal-dual subproblem simplex method. The details of the method is described in Algorithm 3.2 and a sample figure is illustrated in Figure 3.1. For the convergence of the method, we refer the reader to [64], which

Algorithm 3.1 Primal-Dual Simplex Method

Step 1 Obtain an initial dual feasible solution π, and construct the restricted master problem using columns with zero reduced costs based on π.

Step 2 Solve the restricted master problem using the simplex method. If the optimal solution of the restricted master problem has $\xi_{opt} = 0$, then x and π give the primal and dual optimal solutions. Stop.

Step 3 Otherwise, obtain the dual solution ρ of the restricted master problem. Go to Step 4.

Step 4 Improve the dual feasible solution with π and ρ, i.e. let $\pi' = \pi + \theta\rho$ for $\theta \geq 0$ such that $\pi' A \leq c$. Here $\theta = \min_{j \notin J}\{\frac{c_j - \pi A_j}{\rho A_j} | \rho A_j > 0\}$. If no columns satisfying $\rho A_j > 0$, the dual problem is unbounded and the primal problem is infeasible. Stop.

Step 5 Construct the restricted master problem by keeping the optimal basis from the last restricted master problem and adding a set of columns with zero reduced costs cut-off to form a new restricted master problem. Go to Step 2.

will not be restated here.

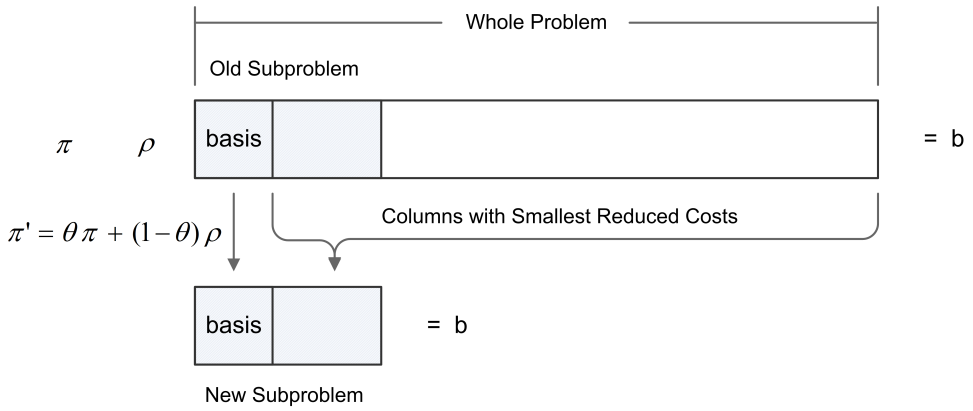

Figure 3.1: Primal-dual subproblem simplex method.

3.4 Extended Pairing Tree for Pricing Out

In the duty tree approach, the set of all legal pairings are stored implicitly in a pairing tree. A pairing tree contains the following components:

- The set of flight F

Algorithm 3.2 Primal-Dual Subproblem Simplex Method

Step 1 Obtain an initial dual feasible solution π, and construct the subproblem using columns with smallest reduced costs based on π.

Step 2 Solve the subproblem using the simplex method, get the optimal primal solution x and dual solution ρ.

Step 3 If ρ is dual feasible for (**D**), then x and ρ give the primal and dual optimal solutions, stop. If $\pi b = cx$, x and π give the optimal solutions, stop. Otherwise, go to Step 4.

Step 4 Improve the dual feasible solution by taking the convex combination of π and ρ, i.e. let $\pi' = \theta\pi + (1-\theta)\rho$ for $0 \leq \theta \leq 1$ such that $\pi' A \leq c$, and $\pi' b$ is as large as possible. Here $\theta = \max\{0, \frac{-\bar{c}_j^{\rho}}{\bar{c}_j^{\pi} - \bar{c}_j^{\rho}} | \bar{c}_j^{\rho} < 0\}$, where $\bar{c}_j^{\pi} = c_j - \pi A_j$, $\bar{c}_j^{\rho} = c_j - \rho A_j$.

Step 5 Price out all columns in the original problem using the improved dual feasible solution $\pi^{'}$, and construct subproblem by keeping the optimal basis from the last subproblem and adding a set of columns whose reduced costs are zero or close to zero, i.e. within small cut-off ϵ, to form a new subproblem. Go to Step 2.

- The set of crew bases CB
- The night connection list for each flight $f \in F$
- The duty tree thread for each flight $f \in F$
- The end duty tree thread for each flight $f \in F$ and each crew base $cb \in CB$
- The subtree mask for each partial pairing contained in the pairing tree
- The maximum number of duties in a pairing

The duty tree approach can store a huge number of crew pairings using a pretty small amount of computer memory. When developing the theory to use the duty tree to solve crew pairing problems, an extended representation of all pairings using more straightforward tree structures is used to illustrate the analysis. Note that the extended representation is purely for illustration purposes because the actual logic and implementation are still following those presented in the previous chapter.

Let $F = \{f_1, f_2, ..., f_m\}$ be the set of flights, $F_{cb} \subset F$ be the subset of flights departing from the crew base, and $D = \{d_1, d_2, ..., f_n\}$ be the set of duties. Let $NC_f \subset F$ be the set of night connection flights for flight f.

Let $D_f \subset D$ be the subset of duties starting from flight f, and $ED_{f,cb} \subset D_f$ be the subset of duties starting from flight f and ending at crew base cb. $\bar{D}_{f,p} \subset D_f$ is the subset of duties for partial pairing p, such that $\{p, d\} \forall d \in \bar{D}_{f,p}$ is a legal pairing or legal partial pairing. $\overline{ED}_{f,cb,p} \subset ED_{f,cb}$ is the subset of duties for partial pairing p, such that $\{p, d\}, \forall d \in \overline{ED}_{f,cb,p}$ is a legal pairing.

The extended pairing tree is represented as follows:

1. Start from the crew base and set it as the root.

2. For each $f \in F_{cb}$, add the set of duties D_f as nodes, and add an arc between the root and each of the nodes. If a duty arrives at the crew base, it is a leaf node. We mark it with a star telling that it cannot be further extended. Such nodes compose the set of 1-duty pairings. Other nodes represent 1-duty partial pairings and will be extended in the following steps.

3. For each unprocessed partial pairing p, find its last flight f, get the night connection list NC_f, and for each flight $i \in NC_f$, add the set of duties $\bar{D}_{i,p}$ as nodes and mark nodes arriving at the crew base as leaves. Add an arc from the node corresponding to partial pairing p to these nodes.

4. Repeat the above step for each unprocessed partial pairing until the maximum number of duties is reached. For a partial pairing p and its last flight f, get the night connection list NC_f. For each flight $i \in NC_f$, add the set of end duties $\overline{ED}_{i,cb,p}$ as nodes and mark all of them as leaves. Add an arc from the node corresponding to partial pairing p to these nodes.

Figure 3.2 shows an example of the extended pairing tree in which all nodes other than the root are duties. All leaf nodes correspond to a legal pairing, and all non-leaf

nodes correspond to a legal partial pairing. The height of the tree is the maximum number of duties in a pairing, and the last layer of the tree contains all end duties that arrive at the crew base. A duty can be duplicated in different parts of the tree. There is such a tree for each crew base, and each tree contains all legal crew pairings for the corresponding crew base. The union of the trees contains all legal crew pairings for the crew pairing problem, which is often represented in the constraint matrix or A matrix of the set partitioning formulation.

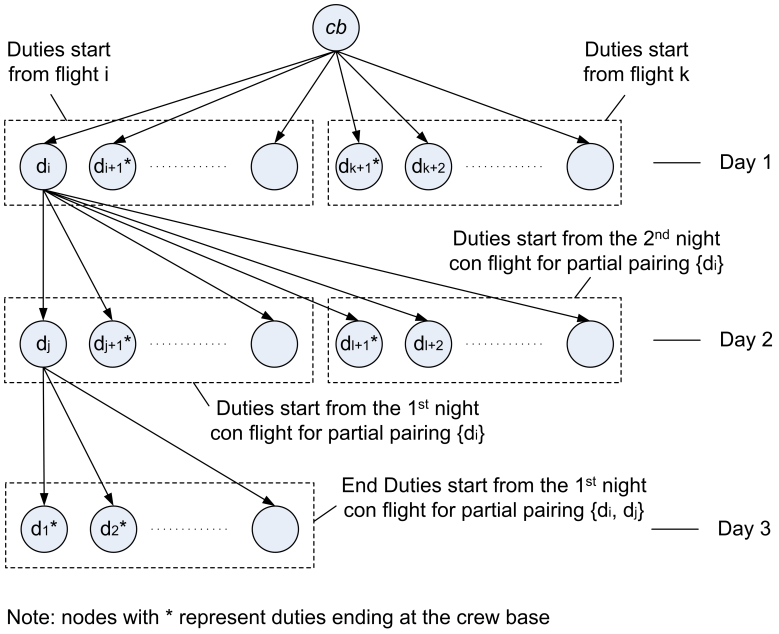

Figure 3.2: The extended pairing tree.

In the extended pairing tree, all duties are explicitly listed for illustration purposes. In the duty tree approach, all duties are implicitly represented in duty trees, end duty trees, and subtrees. Nevertheless, these two types of representations are equivalent in terms of storing the same set of all legal pairings:

1. D_f is the list of duties contained in duty tree for flight f, for all $f \in F$.

2. $ED_{f,cb}$ is the list of end duties contained in the end duty tree for flight f and crew base cb.

3. $\bar{D}_{i,p}$ is the list of valid duties that can be added to partial pairing p contained in the subtree of flight f's duty tree, stored in a mask with the numbers for keeps and skips.

4. $\overline{ED}_{i,cb,p}$ is the list of valid duties that can be added to partial pairing p contained in the subtree of the end duty tree for flight f and crew base cb, stored in a mask with the numbers for keeps and skips.

The only difference between the above representations is that duties are listed explicitly in the extended pairing tree but implicitly in the duty tree approach. In the extended pairing tree, duties are represented next to each other with the same parent such as the root or the corresponding partial pairing p. In the duty tree approach, we can scan through all such duties using a depth-first search. We use the extended pairing tree to simplify the discussion of the pricing out scheme.

3.5 Traverse the Pairing Tree

Traversing the pairing tree in the duty tree approach using depth-first search is equivalent to traversing the extended pairing tree, a more straightforward to illustrate the analysis here.

Figure 3.3 shows a rooted tree starting from a cb and its depth-first search traversal that reads

$$\{d_1 - d_5 - d_6 - d_{11} - d_{12} - d_7 - d_2 - d_3 - d_8 - d_4 - d_9 - d_{10}\} \tag{3.3}$$

with leaf nodes

$$\{d_5, d_{11}, d_{12}, d_7, d_2, d_8, d_9, d_{10}\} \tag{3.4}$$

representing eight pairings in the extended pairing tree, that is, $\{d_1, d_5\}$, $\{d_1, d_6, d_{11}\}$, $\{d_1, d_6, d_{12}\}$, $\{d_1, d_7\}$, $\{d_2\}$, $\{d_3, d_8\}$, $\{d_4, d_9\}$, $\{d_4, d_{10}\}$. So here we have one one-duty pairing, five two-duty pairings, and two three-duty pairings. Given a pairing tree,

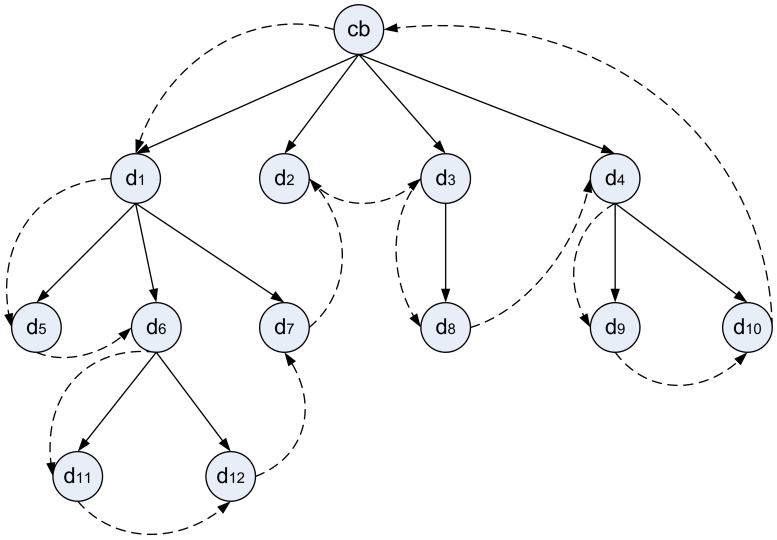

Figure 3.3: The depth first search of a pairing tree.

the order of pairings to be visited during depth-first search is fixed, which is a good advantage when solving crew pairing problems upon the tree.

In addition, there are four partial pairings in this extended pairing tree, i.e. $\{d_1\}$, $\{d_1, d_6\}$, $\{d_3\}$, $\{d_4\}$. As a result, the corresponding duty tree has three sets of subtrees, based on the night connection lists for node d_1, d_3, and d_4, and one set of sub end trees, based on the night connection list for node d_6. Each subtree or sub end tree has a mask to specify which subset of duties in the corresponding duty tree or end duty tree are valid for the corresponding partial pairing.

Similarly, if we traverse the pairing tree in Figure 3.2 using depth-first search, we have pairings $\{d_i, d_j, d_1^*\}$, $\{d_i, d_j, d_2^*\}$,..., $\{d_i, d_{j+1}^*\}$,..., $\{d_{i+1}^*\}$, and so on.

Given a flight $f \in F_{cb}$ for crew base $cb \in CB$, we use depth first search to traverse the pairing tree for flight f. For each duty d contained in the duty list D_f corresponding to the duty tree for flight f, if its last flight arrives at the crew base cb, then duty d is not extended and becomes a 1-duty pairing.

If duty d's last flight arrives at an airport other than the crew base cb, we call it a partial pairing $p = \{d\}$. As introduced in the previous chapter, partial pairings

are not valid pairings because they do not end at the crew base, but they may be extended to a pairing if connected by one or more duties.

To extend partial pairing p, we look at the last flight f_2 of the last duty d in p and get the night connection list N_{f_2} for flight f_2. The night connection list is a set of flights that are legal for a pilot to work on after an overnight rest following flight f_2.

For each flight $f_3 \in N_{f_2}$, there is a duty tree T_{f_3} associated with it, as well as a corresponding subtree because not all duties in the duty tree for flight f_3 can be added to partial pairing p legally as indicated in the previous chapter. The subtree contains valid duties in this duty tree that can legally extend partial pairing p, and is stored as a mask that contains a series number with the the first number specifying valid duties, the second number indicating invalid duties, and so on. The summation of the numbers in the mask is equal to the total number of duties contained in the duty tree T_{f_3}.

We add duty $d \in \bar{D}_{f_3,p}$ from the subtree of the duty tree for flight f_3 to partial pairing p. If the last flight f_4 of duty d ends at the crew base, then we obtain a legal crew pairing $\{p, d\}$. Otherwise, $\{p, d\}$ is still a partial pairing and can be further extended if the total number of duties is less than the maximum number of duties allowed in a pairing as usually specified in a pairing-level legality rule. Using a depth-first search to traverse the whole pairing tree, we can visit all legal crew pairings contained in the pairing tree exactly once.

From set partitioning formulation point of view, traversing the pairing tree is equivalent to processing all columns contained in the constraint matrix once, which is important when calculating the step size in updating dual solutions, calculating the threshold, and selecting the columns with reduced costs less than or equal to the threshold to construct the restricted master problem. All these steps will be discussed in the following sections.

3.6 Calculate Initial Dual π

In the primal-dual subproblem simplex method, we start from an initial dual feasible solution and improve it until a primal and dual feasible solution is found. If the cost function has all positive values, an obvious dual feasible solution is $\pi = 0$, bringing $c - \pi A = c > 0$, with the initial dual objective value being $\pi b = 0$.

For the crew pairing optimization problem that is formulated as a set partitioning problem, it is possible to find a better initial dual feasible solution. For each column in the constraint matrix, we calculate the following ratio:

$$v_j = \frac{c_j}{nz_j}, \ \forall j \in 1, ..., n, \tag{3.5}$$

where c_j is the cost for column j and nz_j is the total number of nonzero elements in column j.

Then the initial dual vector is calculated as

$$\pi_i = \min_{j=1,...,n,\delta_{i,j}=1} v_j, \ \forall i \in 1, ..., m \tag{3.6}$$

where $\delta_{i,j} = \begin{cases} 1, & \text{row } i \text{ has entry 1 at column } j, \\ 0, & \text{otherwise.} \end{cases}$.

That is, we scan through all columns in matrix A, calculate ratios $v_j = \frac{c_j}{nz_j}$ and update the dual values if v_j is smaller. This mechanism is also illustrated in Figure 3.4. The dual vector contains the initialized big values at the beginning of the processing and ends with minimum values from the rows with the above calculation.

To calculate the initial dual solution π in the duty tree approach, we traverse through all pairings using depth-first search and update corresponding values using the above strategy. Because we do not store the intermediate ratio values when we process all columns, no additional storage is needed. The resulted dual solution π will be dual feasible, that is, $c - \pi A \geq 0$, and the dual objective is $\pi b > 0$.

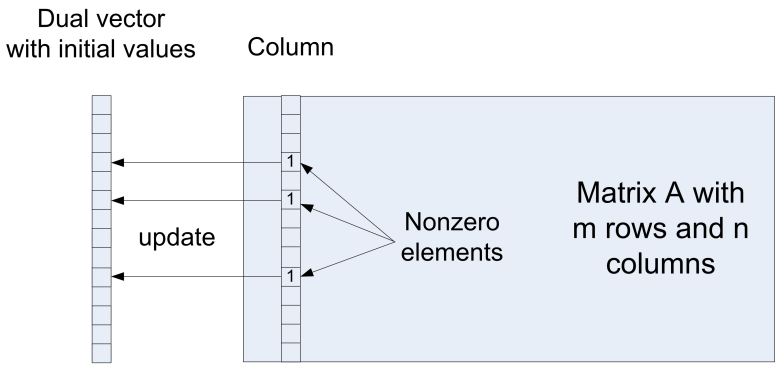

Figure 3.4: Calculate initial dual solution π.

3.7 The Pricing Out Scheme

During the primal-dual subproblem simplex solution process, an important step is to price out. Pricing out is the procedure of calculating reduced costs for all non-basic columns, and in the meantime, calculating the step size for each iteration and generating the subproblem. In this duty tree method, because all pairing columns are implicitly stored, we traverse the duty trees when we price out and generate the subproblem. Because the pairing legality has already been checked, all visited pairings are legal. We can start from each crew base cb and traverse the duty trees T through depth-first search with the strategies discussed in the following sections.

3.7.1 Calculate the Step Size θ in Duty Trees

In the primal-dual subproblem simplex method as discussed in previous sections, at each iteration after obtaining a dual solution ρ for the subproblem, we calculate the step size θ using the formula

$$\theta = \max\{0, \frac{-\bar{c}_j^{\rho}}{\bar{c}_j^{\pi} - \bar{c}_j^{\rho}} | \bar{c}_j^{\rho} < 0\} \tag{3.7}$$

where $\bar{c}_j^{\pi} = c_j - \pi A_j$, $\bar{c}_j^{\rho} = c_j - \rho A_j$.

In the duty tree method, when we compute the step size θ, we traverse the pairing tree using depth-first search for all corresponding columns to do the calculation, that

is, start from each crew base cb, get all flights departing from cb, visit all pairings on the corresponding pairing tree, and for each pairing, use Formula 3.7 to calculate θ. Finally, the minimum θ can be found.

Considering that, during the solution process before a final set of good pairings are selected, the same duty may appear in many different pairings, we propose the following strategy as in Algorithm 3.3 to speed up the pricing out process.

Algorithm 3.3 Speed up the pricing out based on the duty tree

Require: The duty tree T_i for each flight f_i, $f_i \in F$, the masks $M_j, j \in nMask$, where $nMask$ is the number of masks associated with each tree T_i, and the improved dual feasible solution π'.

1: **for** each flight f_i, $f_i \in F$ **do**
2: calculate the reduced cost vector $\bar{c}$ for all duties represented in T_i.
3: **for** each mask M_j, $j \in nMask$ **do**
4: get the reduced cost of the mask from the reduced cost vector $\bar{c}$ based on the pointers pointing to duties in tree T_i.
5: **end for**
6: **end for**

Before pricing out, we first compute the dual value for each duty by summing the dual values of all the flights contained in the duty. During the pricing out process, the duties in each mask can be handled by similar computational efforts to price out a single flight, that is, we can ignore the details of the flight lists involved in the mask, and furthermore, we price out a duty once, and for all the masks that contain this duty, we can take advantage of the same pointer without doing the computation again and again. Thus, we can save the pricing out time significantly.

3.7.2 Calculate the Restricted Master Problem Threshold ϵ

To construct the restricted master problem in the primal-dual subproblem simplex method, we use a reduced cost threshold $\epsilon > 0$. Although it is possible to use a fixed ϵ, the restricted master problem may become too large or too small. If too small, it may take many iterations for the whole pairing problem to eventually converge to

the optimal solution. If too large, the restricted master problem containing too many columns can pose a significant burden on computer memory or make the restricted master problem difficult to solve. In addition, the reduced costs usually decrease over iterations. If we use a fixed threshold, the master problem will become larger and larger.

For large-scale problems, it is often beneficial to use a dynamic threshold so that the size of the restricted master problem can be controlled. The naive way is to collect the reduced costs of all columns, sort them according to their values, and pick the ith element, where i is the desirable restricted master problem size specified by the user. The time complexity of the sort algorithm is $O(n \log n)$. Thus, for large problems, a more efficient way is desirable.

For the duty tree approach, the number of columns can be very large, hundreds of millions or billions or even more. With the good duty tree structure it is possible to store these columns implicitly in the tree. Nevertheless, to explicitly store the reduced costs of these columns can take a large amount of memory.

To find the threshold efficiently without using too much memory, we use a histogram sort variant of the bucket sort. The histogram sort is also known as a counting sort, in which the number of elements in each bucket is counted using an initial pass. Because we do not store elements in buckets, no additional overhead storage memory is needed. Because we are only interested in the threshold, we improve the histogram sort into a histogram kth element algorithm with the detailed procedure as in Algorithm 3.4. Let's denote the estimated minimum reduced cost as $minBucket$ and estimated maximum reduced cost as $maxBucket$ based on the problem at hand.

If the bucket array is large enough and the $maxBucket$ value is selected properly, the threshold value often can be found in one pass according to our computational experiments. We solve a bucket sort problem in each primal-dual subproblem simplex iteration and use the $maxBucket$ value from the previous iterations or can adjust

Algorithm 3.4 Histogram kth element algorithm

Require: The constructed duty trees.

Step 1: Predefine the minimum and maximum restricted master problem size.

Step 2: Create an array of k buckets, and set their initial values to 0.

Step 3: Estimate the minimum and maximum values of the reduced costs, divide the range by k, get the width of the bucket and denote it as w.

Step 4: Calculate the reduced cost for each pairing, and find the bucket index by $i = \lceil \frac{\bar{c}_j - minBucket}{w} \rceil$, where $\bar{c}_j$ is the reduced cost of the jth pairing. If $i > k$, set $i = k$. Increase the count for bucket i by 1.

Step 5: Start from the first bucket, add up the counts for buckets until the total of the cumulative counts falls inside the range of the restricted master problem size. If, at bucket i, the total count is still below the minimum restricted master problem size and, at bucket $i + 1$, the total count is more than the maximum restricted master size, then we set the $maxBucket$ value to $(i+1) \times w$, and repeat the above steps.

Step 6: If there is a bucket i with a total cumulative count within the range for the restricted master problem size, we have found the threshold $\epsilon = minBucket + i \times w$.

accordingly.

Figure 3.5 shows an example of the bucket sort approach for calculating the threshold ϵ. We traverse through the pairing tree to visit every pairing for the reduced costs and count towards the corresponding buckets. Then, we cumulatively add bucket counts until the desirable number of pairings under a certain threshold is found, and set the value corresponding to the bucket as ϵ. Note that the resulting restricted master problem size may not be exact. If more accurate thresholds are needed, the bucket sort algorithm can be run again for the target bucket.

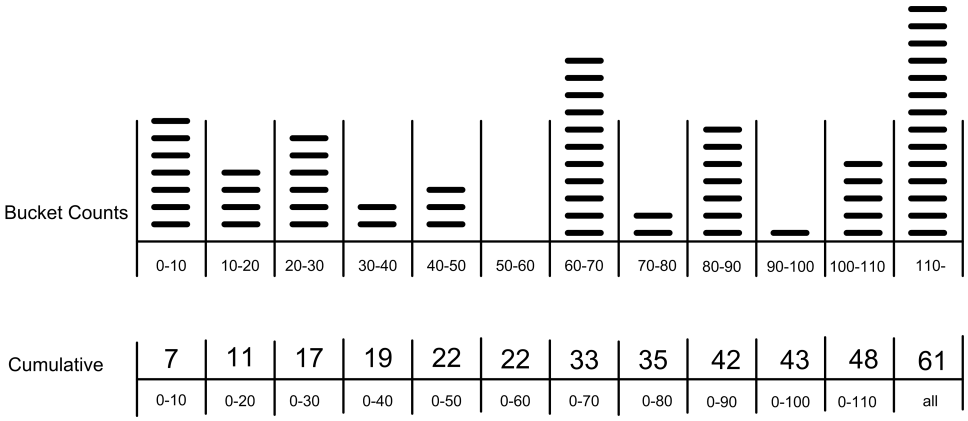

Figure 3.5: Use bucket sort to find threshold ϵ.

Figure 3.6 shows an example of fine tuning the threshold. Suppose we use the bucket sort and find the threshold to be $\epsilon = 70$, the resulting number of columns is 33, while if we reduce the threshold to $\epsilon = 60$, the number of columns will be reduced to 22. We can run the bucket sort again between 60 and 70 if we are not satisfied with either of these two thresholds. The resulting buckets give us the number of columns as 30 with threshold $\epsilon = 65$. This procedure can be carried out recursively until a satisfying threshold is found with the desirable restricted master problem size.

3.7.3 Traverse the Duty Tree to Generate the Subproblem

After obtaining the improved dual feasible solution π', to construct a new subproblem, we need to calculate the reduced costs for all columns and find a set of columns whose

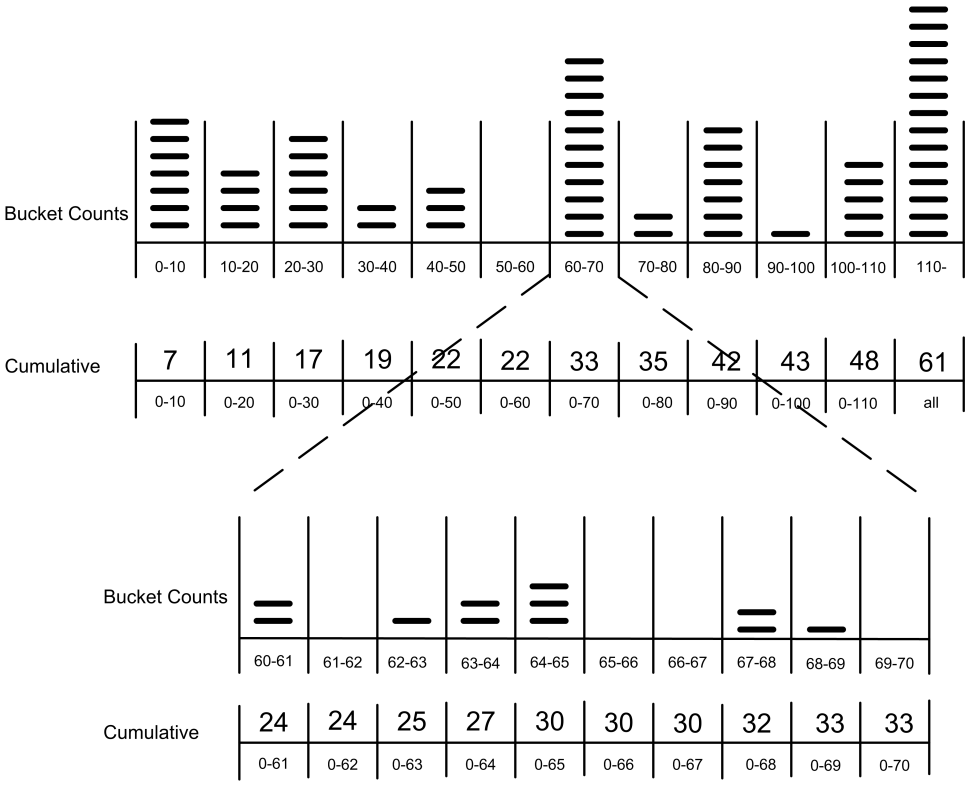

Figure 3.6: Fine-tuning threshold ϵ.

reduced costs are zero or close to zero but within the small cut-off threshold ϵ. That is, with the feasible dual π' and the threshold ϵ, we calculate the reduced cost for each column during the traverse of the pairing tree. When we find a column whose reduced cost is smaller than or equal to ϵ, we add it into the restricted master problem.

3.8 Follow-on Fixing

When the primal-dual subproblem simplex method converges with a primal and dual feasible solution, it is global optimal for the linear programming relaxation of the set partitioning problem. However, it is not possible to assign a fractional pairing to a crew member. To solve the crew pairing problems, we need a final integer solution. In the meantime, it is often desirable to find a good integer solution within reasonable computational time.

In this present work, a good integer solution is achieved using the follow-on fixing

strategy as discussed in [8]. The follow-on fixing specifies whether two flights should appear consecutively in a pairing or not. Given a linear programming solution, we can construct potential follow-on fixing candidates by combining the solution values for consecutive flights. Suppose a column is $\{f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \rightarrow f_5 \rightarrow f_6\}$ and its solution value is $x_1 = 0.6$, we have the following consecutive flight pairs: $f_1 \rightarrow f_2 - (x_1)$, $f_2 \rightarrow f_3 - (x_1)$, $f_3 \rightarrow f_4 - (x_1)$, $f_4 \rightarrow f_5 - (x_1)$, $f_5 \rightarrow f_6 - (x_1)$. Suppose we have another column $\{f_1 \rightarrow f_3 \rightarrow f_4 \rightarrow f_6\}$ and its solution value is $x_2 = 0.4$, we have the following consecutive flight pairs: $f_1 \rightarrow f_3 - (x_2)$, $f_3 \rightarrow f_4 - (x_2)$, $f_4 \rightarrow f_6 - (x_2)$. We go through all columns in the final basis with positive solution values and analyze all flight pairs.

For the flight pairs with the same flights, we combine their solution values. For example, we have $f_3 \rightarrow f_4 - (x_1 + x_2)$, $f_1 \rightarrow f_2 - (x_1)$, $f_2 \rightarrow f_3 - (x_1)$, $f_4 \rightarrow f_5 - (x_1)$, $f_5 \rightarrow f_6 - (x_1)$, $f_1 \rightarrow f_3 - (x_2)$, $f_4 \rightarrow f_6 - (x_2)$. We pick the pair $f_3 \rightarrow f_4 - (x_1 + x_2)$ and add a follow-on fixing for flights f_3 and f_4 because $x_1 + x_2 = 1$, the biggest among all pairs.

That is, we will scan through all columns in matrix A, and remove those columns with f_3 but not f_4, f_4 but not f_3, or with f_3 and f_4 but they are not consecutive. Therefore, the remaining columns have either consecutive f_3 and f_4 or no flight f_3 or f_4 at all.

When we apply the follow-on fixing to duty trees, we can use duty tree features to make the process more efficient with the following cases.

1) If the follow-on fixing occurs within a duty tree, end tree or subtree, we can skip duties violating the follow-on fixing, thus avoiding calculating their dual values, and so on.

2) If the follow-on fixing occurs between duties, that is, the follow-on fixing falls between the last flight in a partial pairing and one of the flights in its night

connection list. Clearly, only the follow-on fixing pair is valid, and other flights in the night connection list can be skipped.

3) If the night connection list contains the second flight of a follow-on fixing, and the last flight in the corresponding partial pairing is not the first flight of the follow-on fixing, then this night connection can be skipped.

3.9 Computational Results

We used a similar set of crew pairing problems as discussed in Chapter 2 for some computational runs. Again, the machine used is Intel Core2 Quad CPU Q8300 @ 2.50GHz, RAM 8.00GB, Windows7 Home Premium 64-bit Operating System. During the solution process, we use follow-on fixing followed by a MIP to get the final integer solution. Some detailed computational results are listed in Table 3.1, Table 3.2, and Table 3.3.

Table 3.1: Results for 350-flight problem based on duty trees.

Features	350-flight problem
Number of flights	350
Total number of pairings generated	3,303,013
Total run time for building duty trees (sec)	39.597
Number of P-D subproblem iterations	16
Average subproblem size	10000
Total run time for calculating θ (sec)	93.712
Total run time for calculating ϵ (sec)	90.859
Total run time for generating subproblem (sec)	97.870
Total run time for LP relaxation (sec)	327.723
LP solution	41687.57
Number of follow-on iterations	5
Total run time for follow-on (sec)	287.044
Total run time for MIP (sec)	2246.12
Integer solution	41731.2857
Total run time (sec)	2900.49

The run for the 350-flight problem resulted in 3,303,013 pairings. The primal-dual subproblem simplex process takes 16 iterations with an average subproblem size of 10,000. The total running times for the 16 iterations to calculate the ratio, calculate the reduced cost for tolerance cutoff, and price out for subproblems take similar amounts of time at 93.712, 90.859, and 97.870 seconds each. The total running time for LP relaxation with 16 iterations is 327.723 seconds. The follow-on fixing takes 5 iterations with a total running time of 287.044 seconds. Then, we solve a MIP and get the final integer solution 41731.2857. Since the total block time for the 350-flight problem is 41625, the pay-and-credit is (41731.2857 - 41625)/41625 = 0.2553%. The total computational time to solve the problem from the beginning to the end is 2900.49 seconds, with most time spent on the MIP stage that takes 2246.12 seconds.

Table 3.2: Results for 212-flight problem based on duty trees.

Features	212-flight problem
Number of flights	212
Total number of pairings generated	5,052,622
Total run time for building duty trees (sec)	80.745
Number of P-D subproblem iterations	7
Average subproblem size	10000
Total run time for calculating θ (sec)	52.858
Total run time for calculating ϵ (sec)	63.379
Total run time for generating subproblem (sec)	62.258
Total run time for LP relaxation (sec)	201.845
LP solution	21090.4323
Number of follow-on iterations	10
Total run time for follow-on (sec)	166.324
Total run time for MIP (sec)	1483.09
Integer solution	21250.4286
Total run time (sec)	1932.02

We also tried another problem with 212 flights and 5,052,622 pairings. The LP using primal-dual subproblem simplex process takes 7 iterations with a total running time of 201.845 seconds. The total running time for pricing out is 52.858 + 63.379

+ 62.258 = 178.495 seconds. After 10 follow-on iterations, we solved a MIP to get the final integer solution 21250.4286. The total running time is 1932.02 seconds. Since the total block time is 20121, the pay-and-credit is (21250.4286-20121)/20121 = 5.61%.

Table 3.3: Results for 219-flight problem based on duty trees.

Features	219-flight problem
Number of flights	219
Total number of pairings generated	75,982,683
Total run time for building duty trees (sec)	1757.1
Number of P-D subproblem iterations	14
Average subproblem size	10000
Total run time for calculating θ (sec)	1608.91
Total run time for calculating ϵ (sec)	1858.74
Total run time for generating subproblem (sec)	1574.3
Total run time for LP relaxation (sec)	5197.49
LP solution	22755.82178
Number of follow-on iterations	54
Total run time for follow-on (sec)	31344.3
Total run time for MIP (sec)	276.308
Integer solution	22948.1429
Total run time (sec)	38575.2

For the run with 219 flights and 75,982,683 pairings, it takes 14 primal-dual subproblem simplex iterations to solve the LP relaxation with the run time being 5197.49 seconds. The total run time for pricing out is 1608.91 + 1858.74 + 1574.3 = 5041.95 seconds. The MIP is solved after 54 follow-on iterations, and the final integer solution is 22948.1429. With the total block time being 22039, the pay-and-credit is (22948.1429 - 22039)/22039 = 4.1252%.

3.10 Summary

This chapter included detailed discussion of the mathematical formulation for the crew pairing optimization problem, the primal-dual subproblem simplex method and

many strategies tailored to the solution process for finding promising pairings upon the duty trees, together with several computational runs. The approach is effective as can be seen from the computational results.

CHAPTER IV

THE OPTIMIZATION-BASED APPROACH FOR CAPACITATED VEHICLE ROUTING PROBLEMS

4.1 Introduction

The vehicle routing problem (VRP) considers the distribution of goods between depots and customers. VRP focuses on the determination of a set of routes, each to be performed by a single vehicle starting and ending at its own depot, such that the transportation cost is minimized while all customer requirements are fulfilled and all operational constraints are satisfied. In 1959, Dantzig and Ramser [41] introduced VRP, which is a truck dispatching problem focused on the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a number of service stations supplied by the terminal. The goal of the problem is to minimize the total mileage of the fleet when assigning stations to trucks so that station demands are satisfied. The detailed solution with four routes is illustrated in Figure 4.1. In 1964, Clarke and Wright [32] proposed a greedy heuristic on this subject. Following their work, many models and heuristics were proposed to get optimal or approximate solutions for different variants of the VRP, for example, the VRP with time windows, the VRP with backhauls, the VRP with pickup and delivery, and so on. [110]. The VRP is closely related to two difficult but well-studied combinatorial problems: the traveling salesman problem with the vehicle capacity set to ∞, and the bin packing problem with the edge cost set to 0. The bin packing problem associated with the CVRP is to determine the minimum number of bins, each with capacity $\mathcal{C}$, to load all the n customer items, each with demand $d_i \geq 0, i = 1, ..., n$. Because of its practical relevance and considerable difficulty, VRP is one of the most challenging and intensively studied combinatorial optimization problems.

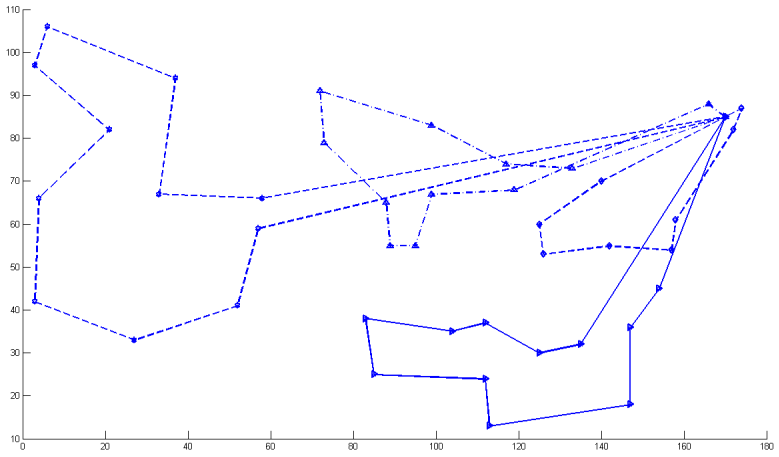

Figure 4.1: CVRP problem introduced by Dantzig in 1959.

4.2 Literature Review

In this research, we focus on the capacitated vehicle routing problem (CVRP), a basic version for many other variants of VRP [110] as shown in Figure 4.2.

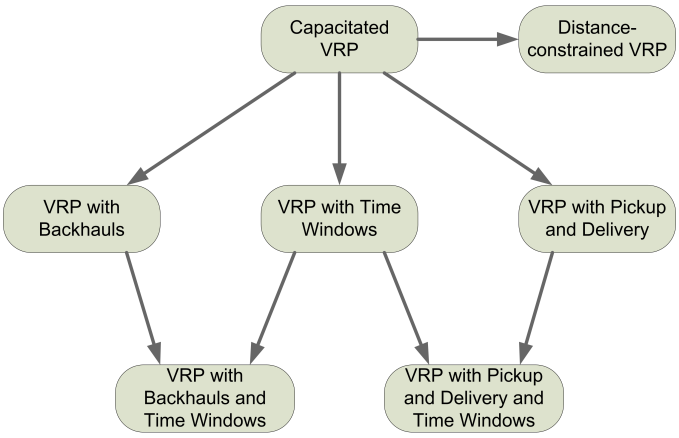

Figure 4.2: CVRP as a basic version of the VRP family.

The CVRP imposes the constraint of limiting the total load of a vehicle along each route to the vehicle capacity, which can be the maximum weight or volume that the vehicle can load. The variant distance-constrained CVRP takes the maximum length or time constraint into consideration.

The VRP with backhauls is an extension of the CVRP. It has two customer subsets with the first subset being the linehaul customers who need the delivery service and the second subset being the backhaul customers who need the picking up service. The problem requires that all linehaul customers must be served before any backhaul customer may be served.

The VRP with time windows, another variant of the CVRP, has both the capacity constraint and a time window $[a_i, b_i]$ associated with each customer i. The time when the vehicles leave the depot, the travel time t_{ij} for each arc $(i, j) \in A$, and each customer's service time s_i are provided. The service of each customer i must start within the time window $[a_i, b_i]$, and the vehicle must stop at the customer location for s_i.

For the VRP with pickup and delivery, each customer i must be served by delivery d_i and pickup p_i with homogeneous commodities, and it is assumed that, at each customer location, the delivery service is performed before the pickup service.

The VRP with backhauls and time windows or VRP with pickup and delivery and time windows is the further extension of the above problems by taking time windows into consideration. Recently, some other variants have been investigated by researchers around the world, and the details will not be listed here.

The CVRP is a generalization of the traveling salesman problem (TSP), which has been well studied as can be seen from [9] and [36]. Many heuristic and metaheuristic approaches are presented in the literature. Rochat and Taillard [96] presented a probabilistic diversification and intensification in local search for VRP. Toth and Vigo [110] provided a good survey on this subject up to 2002. Other recent research includes genetic algorithm [27] and simulated annealing [107]. Franceschi et al. [51] proposed a refinement heuristic approach for distance-constrained CVRP in an attempt to improve on existing heuristic solutions in literature. They discussed a procedure to generate a number of new sequences for the extracted nodes as well as an integer

linear programming model for the re-allocation of some of these sequences to find any possible improvement. A similar work is also discussed in [108].

Other heuristics efforts include a hybrid approach by Marinakis et al. [83], a self-adaptive local search for classical VRP by Alabas-Uslu and Dengiz [6], and an artificial bee colony heuristic by Szeto et al. [106], among others. A parallel and serial algorithm that combines a heuristic local search improvement procedure is provided by [58] and Groër [59]. Another topic for CVRP study is the work by Rodríguez and Ruiz [97], who investigated the effects of many factors such as asymmetry, geographical location of the depot and clients, demand, territory, and maximum vehicle capacity on the solutions of the CVRP for both heuristics and metaheuristics. More recently, the adaptive large neighborhood search heuristic for another variant of the CVRP, the cumulative CVRP, was presented by Ribeiro and Laporte [95], in which the objective is the minimization of the sum of arrival times at customers, instead of the total routing cost. Such problems often arise when the priority is given to satisfy the customer needs, for example, rescue events. In addition, a memetic method was also proposed by Ngueveu et al. [89].

A variety of exact methods are also proposed by different researchers. Toth and Vigo [110] presented a survey for the VRP with detailed analysis of various exact methods, while a specific review for the CVRP can be found in [109] and a more recent one by Baldacci et al. [15]. Some researchers have proposed the branch-and-cut-and-price algorithm, taking advantage of the strengths of both branch-and-cut and branch-and-price techniques. Sample works include branch-and-cut [2], [13], [82], parallel branch-and-cut [92], [93], and branch-and-cut-and-price (a Lagrangian relaxation over q-routes idea combined with the branch-and-cut) by Fukasawa et al. [52].

Furthermore, the set partitioning based algorithms provide another family of exact methods for solving the CVRP. The modeling of the CVRP using a set partitioning

formulation was originally proposed by Balinski and Quandt [16]. A column in the formulation covers a set of customer vertices with total demand not exceeding the vehicle capacity, and has the cost of a minimum route over the depot vertex and the corresponding customer vertices. Later, Agarwal et al. [3] presented a column generation approach based on the set-partitioning formulation, where a single column or several columns are added at each iteration, and the column costs are modified so that the subproblem is solved as a knapsack with the resulting lower bound being used in a branch-and-bound scheme. Hadjiconstantinou et al. [60] presented a branch-and-bound algorithm with the lower bound being computed by solving the dual of the linear programming relaxation of the set partitioning formulation of the CVRP, where the dual solutions are obtained by combining the q-route and k-shortest path relaxations. Baldacci et al. [14] proposed a set partitioning approach with additional cuts and reported improved computing time over the branch-and-cut-and-price approach by Fukasawa et al.[52].

In the literature, promising exact solution approaches for CVRP use various valid linear inequities, such as capacity inequality, comb inequality, multistar inequality, hypotour inequalities, Gomory cuts, and so on. The capacity inequality actually has several that share the same left-hand side but have a different right-hand side: $x(\delta(S)) \geq$ right hand side $, \forall \emptyset \neq S \subseteq \{1, 2, 3, ..., n\}$. The right hand $2\frac{d(S)}{\mathcal{C}}$ gives the fractional capacity inequality, while $2\lceil \frac{d(S)}{\mathcal{C}} \rceil$ produces the rounded capacity inequality, and some other forms generate other types of inequality such as weak capacity inequality, and so on.

If a comb with a handle H and an odd number of teeth $T_1, T_2, ..., T_t$ satisfies such conditions as $H, T_1, T_2, ..., T_t \subseteq V$, $T_j \backslash H \neq \emptyset, \forall 1 \leq j \leq t$, $T_j \cap H \neq \emptyset, \forall 1 \leq j \leq t$, $T_i \cap T_j = \emptyset, \forall 1 \leq i < j \leq t$, and $t \geq 3, odd$, then the comb inequality takes the form as $x(\delta(H)) + \sum_{i=1}^{t} x(\delta(T_i)) \geq 3t + 1$, where $\delta(H)$ denotes the set of edges having one end point in H and one end point not in H, same for $\delta(T_i)$.

For more detailed discussion of different inequalities, the interested reader can refer to the work by Toth and Vigo [110] and Lysgaard et al. [82].

Classically, in cluster-first, route-second method, vertices are first organized into feasible clusters, and for each of them, a vehicle route is constructed. While in route-first, cluster-second method, a tour is first built on all vertices and is then segmented into feasible vehicle routes [110].

The route-first, cluster-second method was first proposed by Beasley [25], in which a giant TSP tour was constructed in the first phase and a shortest path problem on an acyclic graph was solved in the second phase using Dijkstra's algorithm. The cost matrix c_{ij} was defined by:

$$c_{ij} = \begin{cases} d_{0,i+1} + \sum_{k=i+1}^{j-1} d_{k,k+1} + d_{j,0}, & \text{if } i < j, \text{ and route } (0, i+1, ..., j, 0) \text{ feasible;} \\ +\infty & \text{otherwise.} \end{cases}$$

Finally, a partition of the directed giant tour into feasible routes was obtained. Additional information can be found in Prins [91].

In Beasley's method, the resulting vehicle routes are just a part of the giant tour. For example, suppose the giant tour is $0-1-2-3-4-0$. A vehicle route can be $0-1-2-0$, or $0-1-2-3-0$. It cannot be $0-1-4-0$, because skipping customers in the giant tour is not allowed in Beasley's method. In addition, Beasley's method can be applied only to problems without the vehicle constraint because the total number of vehicles cannot be controlled using shortest path algorithms. Moreover, in the literature, the route-first, cluster-second methods are rarely applied by other researchers, and no competitive results compared to other CVRP approaches are reported.

4.3 The Capacitated Vehicle Routing Problems

In this research, we consider the CVRP with the following features:

a) The objective is to minimize the total distance to serve all customers.

b) All customers with delivery requirements and their demands are deterministic and known in advance.

c) The vehicles are identical and based at a single depot.

d) Each vehicle route must start and end at the depot.

e) Each customer is visited by exactly one vehicle route.

f) The sum of the customer demands on each vehicle route does not exceed the vehicle capacity.

From the graph theory perspective, the symmetric CVRP [110] can be represented by a complete and undirected graph consisting of a node set and an edge set. The solution is a set of routes that intersect only at the depot node. An example is illustrated in Figure 4.3, in which nodes a, b, and c represent customers to be serviced, while $c_{ij}, i, j \in \{a, b, c\}, i \neq j$ is the cost associated with each edge traverse, and $d_i, i \in \{a, b, c\}$ is the demand from each customer.

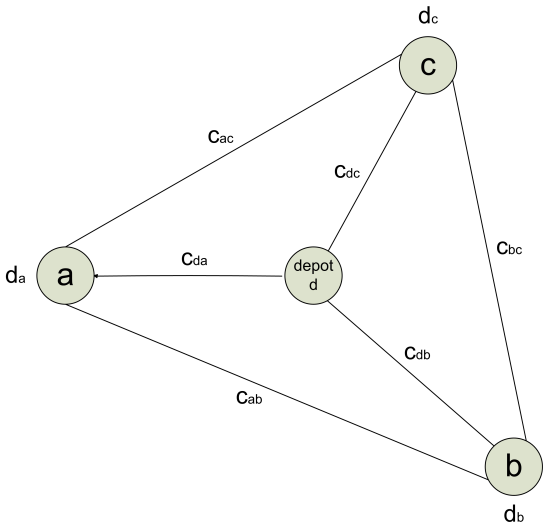

Figure 4.3: A CVRP graph example.

We can construct a network based on the CVRP data by splitting the depot node into two dummy nodes: the source node s and the sink node t. The two dummy nodes s and t represent the same depot, and other nodes represent the customers to be served. We convert the undirected graph into a directed network. Let the constructed network be denoted as $G = (V, A)$, where the vertex set $V = \{s\} \cup \{t\} \cup \{1, 2, ..., n\}$, and arc set $A = (i, j), i \neq j, i \in V, j \in V$. Each arc is associated with a rounded Euclidean distance, that is, the Euclidean distance is rounded to the nearest integer. Any feasible route is represented by a path from the source node s to the sink node t in the underlying network. Figure 4.4 illustrates a small example of the network with customers a, b, and c. $c_{ij}, i, j \in \{s, a, b, c, t\}, i \neq j$ are the traverse costs, and $d_i, i \in \{a, b, c\}$ are the customers' demands.

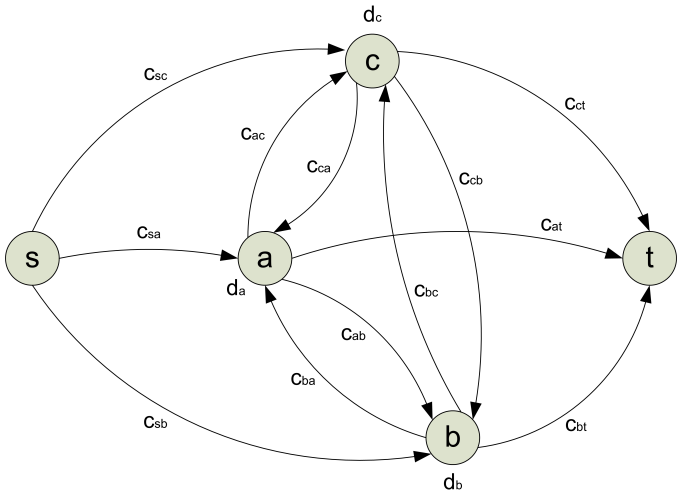

Figure 4.4: An example of the CVRP network.

The following basic notations are introduced for further discussion:

$\{0, 1, 2, ..., n\}$**:** $\{0\}$ represents the depot, and $\{1, ..., n\}$ represent the customers to be served.

c_{ij}**:** The traversal cost between any two customer locations, and in this work we are considering the distance traveled.

$\mathcal{C}$**:** The capacity of each vehicle.

d_i**:** The known and nonnegative demand to be delivered to each customer i, $i = 1, ..., n$. The depot has demand $d_0 = 0$. It is assumed that $d_i \leq \mathcal{C}$, $i = 1, ..., n$, to ensure feasibility of the problem. For the cases when $d_i > \mathcal{C}$ for some customer i, the VRP with split delivery is usually solved where multiple visits to these customers are allowed.

$d(S) = \sum_{i \in S} d_i$ **:** The total demand of the set S, $S \subseteq V \backslash \{0\}$. A trivial bound on the minimum number of vehicles to serve the customers in a set $S = V_u \backslash \{0\}$ can be determined by $\lceil \frac{d(S)}{\mathcal{C}} \rceil$.

$\mathcal{K}$**:** The number of identical vehicles available at the depot. We assume that $\mathcal{K}$ is greater or equal to the minimum number of vehicles $\mathcal{K}_{\min}$ needed to serve all the customers.

Given the coordinates of two customer nodes, for example, (x_i, y_i) and (x_j, y_j), the distance value is determined by the distance between two points. There can be different distance functions as described in the TSPLIB [94] as in Table 4.1, where *nint* indicates the nearest integer.

Table 4.1: Distance functions.

Distance Type	Function
Euclidean distance	$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
Rounded Euclidean distance	$nint(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})$
Manhattan distance	$nint(\lvert x_i - x_j \rvert + \lvert y_i - y_j \rvert)$
Maximum distance	$max(nint(\lvert x_i - x_j \rvert), nint(\lvert y_i - y_j \rvert))$

4.4 Elementary Shortest Path Problem with Resource Constraints

In the set partitioning models for CVRP, because the number of possible routes can be very large, a column generation scheme has been proposed by many researchers. Mourgaya and Vanderbeck [88] discussed a column generation based heuristic for vehicle routing problem, in which the pricing subproblem is solved heuristically. Desaulniers [43] discussed the handling of vehicle routing and crew scheduling problems solved by column generation. At each iteration of the solution process, the restricted master problem is solved by the simplex algorithm, and the subproblem is solved as an elementary shortest path problem with resource constraints (ESPPRC), in which the arc costs represent the reduced costs and the path cost is the reduced cost of the corresponding route variable. Before starting the next iteration, one or several variables with negative reduced costs are added to the restricted master problem.

ESPPRC was first studied by Beasley and Christofides [26]. The ESPPRC finds a shortest path in a graph where the accumulated demand is constrained and no nodes can be visited more than once. If the network is acyclic or the arc costs are positive, the ESPPRC problem can be solved in pseudo-polynomial time. For acyclic networks, it is impossible to form a cycle in the shortest path, and every shortest path is by default elementary. For networks with positive arc costs, the cost of a cycle is always positive, thus, it is impossible for a cycle to stay in a shortest path because the shortest path can be further reduced by removing the cycle. Therefore, the ESPPRC problem is equivalent to a shortest path problem with resource constraints (SPPRC), if the network is acyclic or the arc costs are positive. The SPPRC is weakly NP-hard and can be solved using pseudo-polynomial time algorithms.

However, the pricing network for the CVRP problem is cyclic with possible negative arc costs. This is because dual values for nodes can be either positive or negative, and the arc cost (reduced cost) is negative if the dual value for an arc is larger than

the arc traverse cost $c_{(i,j)}$, that is, $\pi_i > c_{(i,j)}$ and $\bar{c}_{(i,j)} = c_{(i,j)} - \pi_i < 0$.

If the network is cyclic with negative arc costs, the ESPPRC problem is NP-hard in the strong sense [47]. We define the ESPPRC problem as follows: Let $G = (V, A)$ be a graph with the node set V and the arc set A, let $c_{(i,j)}$ be the traverse cost for arc $(i, j) \in E$, let d_i be demand for node $i \in V$ and $\mathcal{C}$ be the corresponding capacity, and let $s \in V$ and $t \in V$ be the source and sink node, the ESPPRC problem is to find path p from s to t with minimum cost and $\sum_{i \in p} d_i \leq \mathcal{C}$

The mathematical formulation for the ESPPRC problem is as follows [69]:

$$\min \sum_{(i,j) \in E} c_{i,j} x_{i,j} \tag{4.1}$$

subject to:

$$\sum_{i \in V} x_{s,i} = 1 \tag{4.2}$$

$$\sum_{i \in V} x_{i,t} = 1 \tag{4.3}$$

$$\sum_{j \in V} x_{i,j} = y_i, \forall i \in V \backslash \{s, t\} \tag{4.4}$$

$$\sum_{j \in V} x_{j,i} = y_i, \forall i \in V \backslash \{s, t\} \tag{4.5}$$

$$\sum_{i \in V} d_i y_i \leq \mathcal{C} \tag{4.6}$$

$$x(E(S)) \geq y(S) - y_i, \forall i \in S, S \subset V, \|S\| \geq 2 \tag{4.7}$$

$$x_{i,j} = \{0, 1\}, \forall (i, j) \in A \tag{4.8}$$

$$y_i = \{0, 1\}, \forall i \in V \backslash \{s, t\} \tag{4.9}$$

$$\tag{4.10}$$

where $x_{i,j}$ is the flow on arc $(i, j) \in A$, y_i is the flow on node $i \in V$, $E(S) = \{(i, j) : i \in S, j \in S\}$, $x(E(S)) = \sum_{(i,j) \in E(S)} x_{i,j}$, $y(S) = \sum_{i \in S} y_i$. Constraints (4.2) and (4.3) are the flow from the source and sink. Constraints (4.4) and (4.5) are flow

conservation constraints. Constraints (4.6) are for resources. Constraints (4.7) define connectivity and cycle elimination, which have an exponential number of constraints.

In the literature some work has been presented to address ESPPRC in VRP, for example, the work by Feillet [50] in which the proposed approach is tested on the VRPTW. Another work on the ESPPRC for the CVRP is by Salani [102]. Salani used many heuristics together with the ESPPRC to obtain a good initial solution and speed up the pricing out process. Nevertheless, bigger benchmark instances still can not be solved within a reasonable computational time [102]. Jepsen, Petersen, and Spoorendonk [69] proposed a branch-and-cut approach for the ESPPRC by relaxing Constraints (4.7). The generalized subtour elimination constraints, 0-1 knapsack cover inequalities, and generalized capacity inequalities are used to find optimal solutions. They reported improved solutions over labeling algorithms.

The ESPPRC is very difficult to solve, so many approaches for VRP formulated over cyclic graphs are relaxed to the shortest path problems with resource constraints (SPPRC) [44], [45].

The SPPRC is an extension of the classic shortest path problem. From the underlying network, the SPPRC is to find a shortest path among all paths starting from a source node and ending at a sink node, satisfying a set of resource constraints.

SPPRC does not require the path to be elementary and has been widely used by approaches based on column generation that are being applied to variants of VRPs and airline crew scheduling problems. The standard approach to solve the SPPRC in practice is based on dynamic programming and has a pseudo-polynomial complexity. The principle is to associate with each possible partial path a label indicating the consumption of resources and to eliminate labels with the help of dominance rules [50]. A classification and a generic formulation for the SPPRCs are recently discussed by Irnich and Desaulniers [68].

However, this relaxation sometimes leads to weak lower bounds and possibly impractical large branch-and-bound trees when solving the VRP using branch-and-price [68]. When solving the pricing subproblem using SPPRC by dropping the constraint that the path must be elementary, the computing time may be less at the cost of yielding weak lower bounds, and the columns may include cycles.

Some studies take the compromise between solving the SPPRC and the ESPPRC, that is, the SPPRC with k-cycle elimination, by forbidding cycles with length k or less. The idea was applied to VRPTW by Kolen et al. [78] and Desrochers et al. [44]. A further extension by increasing the k value to a bigger number than 2 was investigated by Irnich and Villeneuve [67], in which pseudo-polynomial labeling algorithms were analyzed, particularly with $k = 3, 4$, and the computational results tested in VRPTW showed that there is a trade-off between eliminating cycles with longer lengths within each branch-and-bound node and solving more tree nodes. It is reported that the cycle elimination for small values of k can substantially improve the lower bound of the master problem and the tighter lower bound is expected to lead to a smaller branch-and-bound tree [68]. Nevertheless, when the time windows of the VRPTW are wide, the ESPPRC continues to be difficult to solve [71]. The k-cycle elimination approach is applicable more to VRPTW, rather than CVRP, which is more difficult because the time window is ∞.

4.5 The Giant Tour based Mathematical Formulation for CVRP

A giant tour for CVRP is the traveling salesman tour starting from the depot, visiting all the customers, and going back to the depot. There are many different ways to form this tour, for example, from the TSP, from any existing solutions, and so on. In this research, we are using the giant tour as precedence constraints.

Given a giant tour, we have the two-index formulation as in the following.

$$
\begin{aligned}
\min \quad & \sum_{\{i,j\}\in\hat{E}} d_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{\{i,j\}\in\delta(\{h\})} x_{ij} = 2, \forall h \in V\backslash\{0\} \\
& \sum_{\{i,j\}\in\delta(S)} x_{ij} \geq 2k(S), \forall \emptyset \neq S \subseteq V\backslash\{0\} \\
& \sum_{j\in V\backslash\{0\}} x_{0j} = 2\mathcal{K} \\
& x_{ij} \in \{0,1\}, \forall \{i,j\} \in \hat{E}\backslash\{\{0,j\} : j \in V\backslash\{0\}\} \\
& x_{0j} \in \{0,1,2\}, \forall \{0,j\} \in \hat{E}, j \in V\backslash\{0\},
\end{aligned}
$$

where:

x_{ij} is an integer variable,

E is the edge set over the undirected CVRP graph,

$\hat{E} = \{(i,j) \in E : i \text{ is ahead of } j \text{ in the giant tour}, or, j = 0\}$,

$S \subseteq V\backslash\{s,t\}$,

$\delta(S) = \{\{i,j\} \in \hat{E} : i \in S, j \notin S, or, i \notin S, j \in S\}$,

$d(S) = \sum_{i\in S} d_i$,

$k(S)$ is the minimum number of vehicles of capacity $\mathcal{C}$ needed to service all customers in S.

Because calculating $k(S)$ is equivalent to solving a bin packing problem with bin capacity $\mathcal{C}$ and demands in S as item sizes and it is NP-hard, a lower bound such as $\lceil d(S)/\mathcal{C} \rceil$ can be used instead.

If we have multiple giant tours available that can help to achieve better solution quality for the target CVRP, then we take advantage of the set partitioning formulation that can handle inputs from multiple giant tours. Though the set partitioning formulation of the CVRP has a possible exponential number of variables, this model has some attractive features. One is that it is very general because many route constraints can be taken into account when generating feasible routes, which constitute

the columns of the formulation. The other is that the cost of each column can be evaluated separately, allowing consideration of complex cost components. Moreover, the linear programming relaxation is typically very tight [13], and the formulation is simple in form. Thus, in this study, we model the CVRP using the set partitioning formulation. Let $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_r\}$ denote the collection of all feasible routes to the CVRP in target, where a feasible route is one that starts and ends at the depot with $\sum_{i \in \text{route } \mathcal{R}_j} d_i \leq \mathcal{C}$. We have the model as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{j \in R} c_j x_j \\
\text{s.t.} \quad & \sum_{j \in R} a_{ij} x_j = 1, \ \forall \ i \in V \backslash \{0\} && (4.11) \\
& \sum_{j \in R} x_j \leq \mathcal{K} && (4.12) \\
& x_j \in \{0, 1\}, \ \forall \ j \in R, && (4.13)
\end{aligned}$$

where:

c_j**:** The cost of route $\mathcal{R}_j$.

$R = \{1, 2, ..., r\}$**:** The index set of all feasible routes $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_r\}$ with the giant tour as precedence constraints.

a_{ij}**:** $a_{ij} = \begin{cases} 1, & \text{if route } \mathcal{R}_j \text{ covers customer } i, \\ 0, & \text{otherwise .} \end{cases}$

x_j**:** Decision variable, $x_j = \begin{cases} 1, & \text{if route } \mathcal{R}_j \text{ is selected in the solution,} \\ 0, & \text{otherwise .} \end{cases}$

$\mathcal{K}$**:** The maximum number of vehicles available at the depot. We assume that $\mathcal{K} \geq \mathcal{K}_{\min}$, where $\mathcal{K}_{\min}$ denotes the minimum number of vehicles required to serve all the customers and may be determined by solving the corresponding bin packing problem, that is, calculate the minimum number of bins to load all the n items

with each bin's capacity being $\mathcal{C}$ and each item's weight being nonnegative $d_j, j = 1, ..., n$.

Constraint (4.11) imposes that each customer be visited exactly in one route, and Constraint (4.12) requires that at most $\mathcal{K}$ routes can be used. This formulation has a coefficient matrix with a row for each customer and a column for each route. The entries in the matrix are 0 or 1 with a 1 signifying that the route corresponding to the column includes a visit to the customer corresponding to the row.

To solve the CVRP, we specify the following assumptions:

a) The objective we consider is to minimize the total distance traveled, where the number of vehicles in the solution may not be necessarily minimized. When the objective is to minimize only the number of vehicles, the CVRP reduces to a bin packing problem.

b) The cost structure is assumed to be symmetric, that is, $c_{ij} = c_{ji}, i \neq j$, where i, j denote the customer locations indices. In practical applications, the travel cost structure for the CVRP can either be symmetric when both directions between each pair of customer locations have the same travel cost or be asymmetric when there are some limitations on the road usage such as a one-way direction.

c) The deliveries cannot be split, that is, a customer order cannot be served by two or more vehicles.

d) There is a vehicle capacity constraint, but no travel distance constraints.

In practice, the vehicle capacities $\mathcal{C}_k, k = 1, ..., \mathcal{K}$ may be different or there may have multiple depots, which lead to different variants of the CVRP.

A feasible solution $\mathcal{R}_1, ..., \mathcal{R}_k$ of the CVRP consists of the following:

a) A partition of the customers into k subsets S_i such that $\sum_{j \in S_i} d_j \leq \mathcal{C}$, $1 \leq i \leq k$.

b) A permutation σ_i of $S_i \cup \{0\}$ specifying the service order of the customers in S_i, $1 \leq i \leq k$.

To solve the above formulated problem, its linear programming relaxation is usually solved first by removing the integrality constraint on the x variables. The optimal solution provides a lower bound on the value of the optimal integer solution. The set partitioning model can accommodate additional constraints because these constraints serve only to reduce the number of columns with the legality check. Nevertheless, the approach may result in the number of columns being extremely large.

4.6 Build the Acyclic Capacity Expanded Compact Storage Route Network

It is obvious that the CVRP classic network is cyclic, which makes the pricing out process very challenging because we have negative arc costs that represents the corresponding reduced costs, during the subsequent solution approach with the set-partitioning model.

We used Beasley's idea of using a giant tour to create an acyclic network to avoid negative reduced cost cycles. However, we change the giant tour concept to a customer visiting sequence so that the vehicle route does not necessarily follow the giant tour. As long as customer j is behind customer i in the visit sequence, customer j can follow customer i immediately in a vehicle route.

In this case, Beasley's shortest path approach is no longer applicable. We propose a set partitioning formulation and introduce the expanded route network structure based on a given customer visiting sequence, so that each route network contains all the feasible routes satisfying the capacity constraints. In addition, all the routes in the route network are feasible routes. We can store this huge number of routes in the network, and we know exactly how many feasible routes there for the optimization process when solving the set partitioning model.

4.6.1 Build the Acyclic Network

To build the acyclic network from the initial cyclic network $G = \{V, A\}$, first, we establish the visiting sequence of all customers. A basic definition follows.

Definition 4.6.1. *Visiting Sequence: Given a set of customers $V\backslash\{s,t\}$, we start from the depot, visit each customer exactly once, and return to the depot in one route, without considering the vehicle capacity.*

Let's denote the visiting sequence as VS. In a VS, if node $i, i \in V\backslash\{s,t\}$ is closer than node $j, j \in V\backslash\{s,t\}$ to the source node s, then node i is the ancestor of node j, and node j is the descendant of node i.

In this study, different ways are used for obtaining the visiting sequence, for example, the solution from a traveling salesman problem, the solution from existing literature, and some random permutation upon these solutions, etc. For the details of the traveling salesman problem, the reader is referred to [9] for the theory and algorithms, as well as Concorde [35].

With an established visiting sequence, we can build the acyclic route network structure as in Algorithm 4.1.

Algorithm 4.1 Build the Acyclic Network

1: Given the customer set $S = V\backslash\{0\}$.
2: Establish visiting sequence VS.
3: Follow the visiting sequence in VS, build an arc from a node $i, i \in V\backslash\{s,t\}$ to another node $j, j \in V\backslash\{s,t\}$, where i is the ancestor of j in VS.
4: Build an arc from source s to any customer node.
5: Build an arc from any customer node to sink t.

Figure 4.5 illustrates the acyclic network corresponding to Figure 4.4. In this example, suppose the visiting sequence for the three customers is b, c, and a, then we have arcs from an ancestor node to its descendent node, from the source to all three customer nodes, and from all three customer nodes back to the sink node.

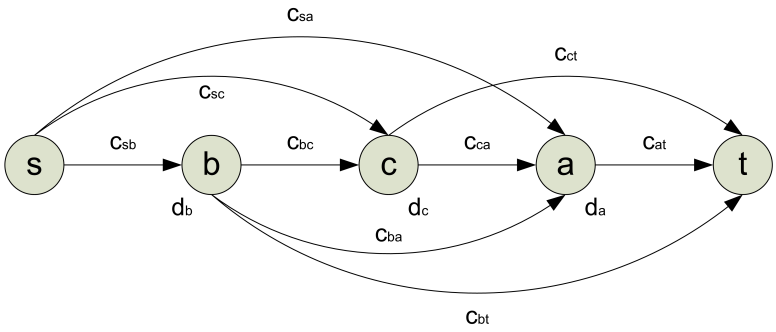

Figure 4.5: The acyclic network example.

In the acyclic network, a feasible solution of the CVRP problem consists of paths from the network, while paths from the network may not be feasible because of the vehicle capacity constraint. In this case, a resource constrained shortest path problem needs to be solved during the pricing out process if we try to find a solution based on this network.

4.6.2 Build Capacity Expanded Compact Storage Route Network

To solve the CVRP, if we have the constraint that the sum of the demands on the feasible path cannot exceed the vehicle capacity, then the paths in the network may not be legal due to this constraint. We propose a method by constructing a capacity expanded compact storage route network. The following concepts will be used in this discussion.

Definition 4.6.2. *Partial route: Starts from the depot and visits a subset of customers without returning to the depot. It can be extended by visiting more customers or completed by returning to the depot.*

Definition 4.6.3. *Load of a partial route: The sum of customer demands on the partial route.*

To create a capacity expanded network, for each customer node, we establish a copy of the node for each capacity layer, that is, $0, 1, ..., \mathcal{C}$. A node is denoted as

$v^{i,k}$ if customer i is added to partial routes with load k. These partial routes can be any partial routes as long as they have a total load of k. For nodes at layer 0, their partial routes contain only the source node s, that is, they are only connected from the depot. So the load at this point is 0. The definition of the capacity expanded route network is given in Definition 4.6.4 and the building blocks are shown in Figure 4.6.

Definition 4.6.4. *Capacity expanded route network $G = (V, A)$: For CVRP problem with n customers, demand $d_i, i = 1, ..., n$, vehicle capacity $\mathcal{C}$, and visiting sequence $\{1, 2, ..., n\}$, We have the node set:*

$$V = \{s, t\} \cup v^{i,c}, i = 1, ..., n, c = 0, ..., \mathcal{C}.$$

That is, for each customer $i = 1, ..., n$, there are $\mathcal{C} + 1$ replicates of nodes $v^{i,c}$, where $c = 0, ..., \mathcal{C}$. Node $v^{i,c}$ indicates the partial route until customer i has total load of c. In addition, there is a source node s and a sink node t corresponding to the depot. And we have the arc set:

$$A = \begin{cases} (v^{i,c_1}, v^{j,c_2}), & \text{for all } 0 \leq i < j \leq n, \text{ and } c_1 + d_i = c_2, \\ (s, v^{i,0}), & i = 1, ..., n, \\ (v^{i,c}, t), & i = 1, ..., n, \ c = 1, ..., \mathcal{C}, \text{ and } c + d_i \leq \mathcal{C}. \end{cases}$$

A node $v^{i,c}, i = 1, ..., n, c = 1, ..., \mathcal{C}$ is valid if there is a path from the source node s to $v^{i,c}$, and $c + d_i \leq \mathcal{C}$. An arc is valid if both its tail and head nodes are valid.

The capacity expanded route network consists of all valid nodes and valid arcs associated with this visiting sequence.

The detailed procedures for building the network are described in Algorithm 4.2.

Figure 4.7 illustrates the acyclic capacity expanded route network. In the figure, the customer nodes are reproduced at each load layer that is increased by 1 at each layer. We have arcs from the depot source node to all of the customer nodes at load

Algorithm 4.2 Build the Capacity Expanded Acyclic Route Network

Require: Depot node as source s, customers to be served $\{1, ..., n\}$, each customer's demand $d_1, d_2, ..., d_n$, and vehicle capacity $\mathcal{C}$.

1: Given a customer visit sequence VS, order customer nodes by the visit sequence. Denote each customer node i as $v^{i,0}$.
2: Reproduce each customer node i by $\mathcal{C}$ copies as $v^{i,1}, v^{i,2}, ..., v^{i,\mathcal{C}}$ so that it has a copy in each of the $\mathcal{C}+1$ load layer, starting from layer 0 for capacity 0 and increasing capacity by 1 for each layer. Node $v^{i,c}$ signifies that the path from the source s until node $v^{i,c}$ has consumed c units of the vehicle capacity $\mathcal{C}$.
3: Build an arc from the depot source node s to all nodes at load layer 0.
4: **if** customer node i is ahead of node j in the customer visit sequence VS **then**
5: **if** node i has customer demand d_i and is in load layer c, node j has customer demand d_j and is in capacity layer $d_i + c$ **then**
6: **if** $d_i + c + d_j \leq \mathcal{C}$ **then**
7: Build an arc between node i and j.
8: **end if**
9: **end if**
10: **end if**
11: Nodes without a path from the source node s are invalid.
12: Build an arc from any valid customer node to depot's sink node t.
13: The cost of a route equals the cost from the depot's source s to sink t with any valid customer node visited along the way.

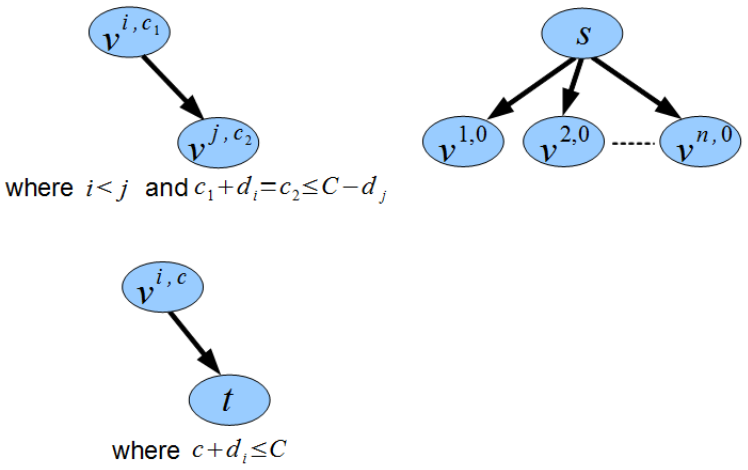

Figure 4.6: Capacity expanded route network building blocks.

layer 0. The arcs from load layer 0 to any other load layer are dependent on customer demands and meet the visiting sequence. For example, we build arcs $(I, J^v), (I, K^v)$, and (I, L^v) because of demand $d_I = v$ and visiting sequence $I - J - K - L$ with customers J, K, and L being visited after I. Other arcs are built in a similar way, and finally, each of the active nodes has an arc going back to the depot sink node.

The route network we build has the following good properties:

Property 4.6.1. *Route Network Properties:*

a) There is a route network for each customer visit sequence.

b) The network is acyclic.

c) Every feasible route of the CVRP is a directed path from the source node s to the sink node t. Conversely, every path from the source node s to the sink node t defines a legal route to the CVRP.

d) The network serves as a compact storage for all legal paths. The legal paths can be enumerated from the network.

The capacity constrained shortest path problem is now reduced to finding a shortest path that begins at the source node s and terminates at the sink node t. Because the network is acyclic and the vehicle capacity constraint has been represented in the

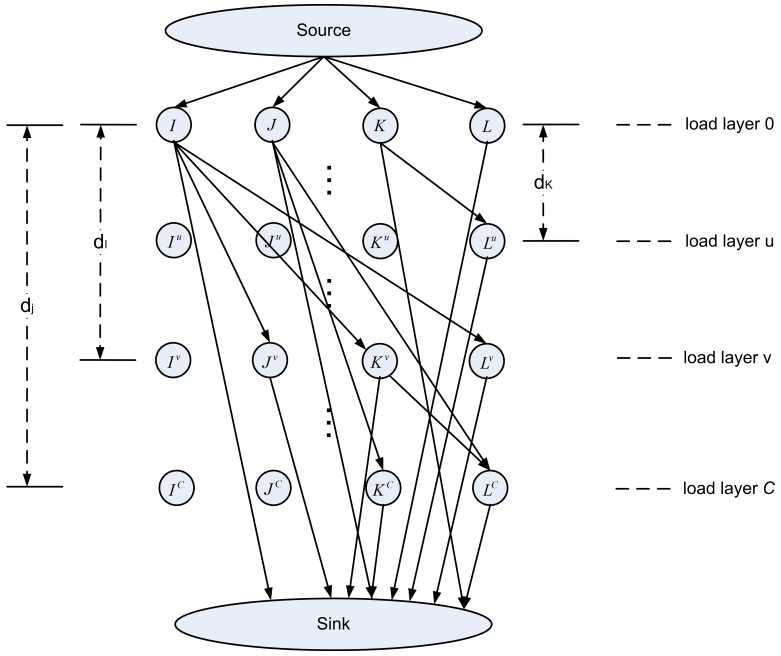

Figure 4.7: Construction of capacity expanded route network.

network, the resulting shortest path problem is efficiently solvable, and all the paths in the network satisfy the capacity constraint.

The resulting compact storage route network is acyclic with the capacity constraint being satisfied, and we can find the shortest path using a linear time reaching algorithm and perform a depth-first search to enumerate the legal paths.

4.6.3 The Enumeration with Tightness Bound

During the enumeration process, we find that, for some CVRP problems, the sum of all the demands $\sum_{i=1,...,n} d_i$ is close to the transportation capability $\mathcal{KC}$ of all the available vehicles. For such situations, we define the concepts of tightness and minimal vehicle load as follows:

$$\text{Tightness } \delta_{\text{avg}} = \frac{\sum_{i=1,...,n} d_i}{\mathcal{KC}}$$

where $\mathcal{K}$ is the number of available vehicles, $\mathcal{C}$ is the vehicle capacity, and $d_i, i = 1, ..., n$ is the demand of each customer.

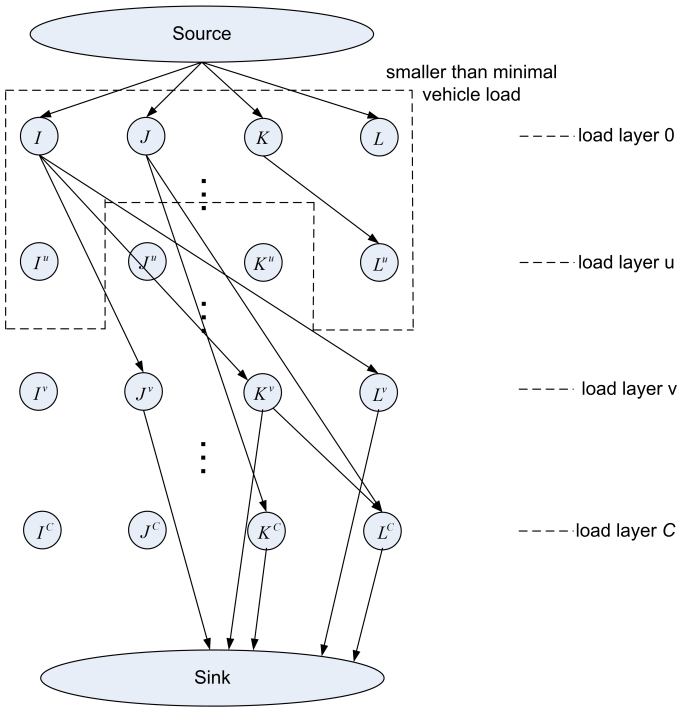

Figure 4.8: CVRP route network with tightness.

The tightness δ indicates the capacity usage of all the vehicles available in the depot in terms of the total demand of the customers that these vehicles are to serve. Then we can compute the minimal vehicle load as in the following:

$$\text{Minimal Vehicle Load } \mathcal{L}_{min} = \mathcal{C} - (\mathcal{K} \times \mathcal{C} - \sum_{i=1}^{n} d_i)$$

Those impossible routes that fall under $\mathcal{L}_{min}$ can be identified and will be removed because they cannot be part of an optimal solution. Thus, we filter out impossible routes, that is, we get the minimal vehicle load $\mathcal{L}_{min}$ from the problem data, check the generated routes against the $\mathcal{L}_{min}$ during the route construction process, and remove those routes identified as being impossible without bringing them into the solution search space. For example, if the load at this point plus $d_i, i \in \{I, J, K, L, I^u, L^u\}$ is less than $\mathcal{L}_{min}$, then we have the route network with tightness as shown in Figure 4.8.

In this figure, the arcs from nodes I, J, K, L, I^u, and L^u back to the sink become

invalid because they violate the minimal vehicle load and, thus, can be removed from the network.

For the benchmark problems, the greater the tightness, the better for enumerating over the minimal vehicle load. In this case, we can eliminate many columns with total demand smaller than the minimal vehicle load because they cannot be feasible candidates for the final solution. For example, if a route has a $\sum_{i \in \mathcal{R}_r} d_i$ smaller than 80 while the minimal vehicle load is 85, we can delete this route from the route list without further consideration. This is a scheme that can reduce the route list size without affecting the solution quality.

4.7 The Primal-Dual Subproblem Simplex Method for CVRP Problems

The CVRP represented on the compact storage route network as discussed in previous sections is formulated as a set partitioning model and solved using the primal-dual subproblem simplex method. The basic algorithm of the method has been discussed in detail in Chapter 3, so we will not restate it here. In the following sections, some features of the primal-dual subproblem simplex method for the CVRP problem are discussed.

4.7.1 The Initial Dual Feasible Solution

As discussed in Chapter 3, $\pi = 0$ can be an initial dual feasible solution because the column cost is the vehicle travel distance for a route, which is positive.

To find a better initial dual solution, we use a repeated shortest path approach to find a dual feasible solution. Given the initial dual values that can be set as large numbers, we find the shortest path. If the shortest path is negative, the ratio is calculated by dividing the route distance by the number of nodes visited in the shortest path. The corresponding dual values associated with the nodes are replaced by the ratio, if it is smaller. This procedure is repeated until the shortest path is

nonnegative, resulting in an initial dual feasible solution. This process is equivalent to the strategy used for the duty tree approach.

4.7.2 The Restricted Master Problem

On the network, the arc cost represents the reduced cost $\bar{c}_{ij} = \text{dist}_{ij} - \pi_i$, with the following features.

Cumulative cost structure: One of the attractive features of the route network is that the cost of a path or a route is cumulative; that is, it is the sum of the cost of the sub-routes on it.

Reduced costs: During the pricing step, we need to find source-sink paths with the smallest reduced costs. On the route network, each out-going arc cost, that is, reduced cost, is calculated as the original arc cost minus the dual vector on the associated node. With the cumulative cost structure, the reduced cost for a path p is exactly the sum of the reduced arc costs on the path. For example, a path $s-1-2-3-t$ visits customer nodes $1, 2$, and 3 in sequence, then we have $\bar{c}_p = (c_{s1} + c_{12} + c_{23} + c_{3t}) - (\pi_1 + \pi_2 + \pi_3) = c_{s1} + c_{12} - \pi_1 + c_{23} - \pi_2 + c_{3t} - \pi_3 = c_{s1} + \bar{c}_{12} + \bar{c}_{23} + \bar{c}_{3t}$

All Legal Paths: Because the capacity constraint has been represented in the route network, all paths from source s to sink t are legal. Thus, the legality check in terms of capacity constraint is not needed during the iterative solution process.

For the pricing process, because the network is acyclic, we can solve a shortest path problem on the acyclic network efficiently using the reaching algorithm. To generate columns with small reduced costs to form the restricted master problem, we enumerate columns from the network, and those columns with reduced costs less than or equal to the threshold ϵ will be used.

4.7.3 Calculate the Dual Update Step Size

After the restricted master problem is solved, we have the optimal dual solution ρ. The new dual feasible solution π' is a convex combination of the current dual feasible solution π and the optimal dual solution ρ from the restricted master problem.

The new dual feasible solution is calculated as follows:

$$\pi' = \theta\pi + (1-\theta)\rho \tag{4.14}$$

A good step size θ is needed while maintaining dual feasibility for π', i.e. $c - \pi' A \geq 0$. According to the previous chapters, it can be calculated as [64]:

$$\theta = \max\{0, \frac{-\bar{c}_j^\rho}{\bar{c}_j^\pi - \bar{c}_j^\rho} | \bar{c}_j^\rho < 0\} \tag{4.15}$$

where $\bar{c}_j^\pi = c_j - \pi A_j$, $\bar{c}_j^\rho = c_j - \rho A_j$, and j is a column from matrix A.

Similarly, we use a repeated shortest path algorithm to calculate the step size θ. First, we set $\theta = 0$, update the feasible dual solution π' using $\pi' = \theta\pi + (1-\theta)\rho = \rho$, and find the shortest path. If the shortest path is nonnegative, $\theta = 0$, and ρ is dual feasible for the problem. Thus, we have found the optimal solution.

Otherwise, a new step size $\theta = \frac{-\bar{c}_j^\rho}{\bar{c}_j^\pi - \bar{c}_j^\rho}$ is calculated, where $0 \leq \theta \leq 1$ and column j is from the shortest path. With the new θ, we update π by $\pi' = \theta\pi + (1-\theta)\rho$ and repeat the above process. We have found a good dual step size θ if the shortest path is nonnegative.

We update the dual feasible solution using this step size θ, and the dual feasible objective is improved accordingly. Then we move to the next primal-dual subproblem simplex iteration.

4.7.4 Follow-on Fixing

After the linear programming relaxation is solved to optimality, we use follow-on fixing to find an integer solution. The follow-on fixing pairs, in which two customers

will be visited consecutively using the same vehicle, are generated the same way as in Chapter 3, based on the optimal solution from the linear programming relaxation.

To use the follow-on fixing to reduce the network, we check all arcs against the follow-on pairs. If customer i and j are the follow-on, we disable all arcs with tail node i and head node other than j. Similarly, we disable all arcs with head node j and tail node other than i. After such arcs are disabled, all routes generated have i and j being visited consecutively, or neither of them present.

4.8 Solution Framework with Multiple Route Networks

The bottleneck of the set partitioning formulation is the pricing out step, which solves an ESPPRC problem. The ESPPRC problem is difficult because the CVRP network is cyclic. If we know the customer visit sequence, then the network will become acyclic, and the elementary shortest path problem can be solved in polynomial time. Based on this observation, we proposed a giant tour based method for the CVRP problem. In this approach, several possible customer visit sequences will be used as input, such as the sequence from the current best known solution, the TSP sequence, or other random permutations. For each customer visit sequence, we construct an acyclic route network based on the sequence and store all legal routes. Then, we solve a set partitioning problem using the columns stored on these route networks. This approach is an approximation because these route networks contain only a subset of all legal routes for the CVRP problem, so the solutions may be suboptimal. However, because the current best known sequence is included as the input, this algorithm can find a solution as good as or better than the current best known solution.

During the search for solution improvement, we also construct multiple initial solutions and build their corresponding networks. Then, at each iteration, we get many columns with negative reduced costs within a certain threshold and add them into the subproblem columns to determine possible improvement or better solutions

than currently found in the literature for those published CVRP problems.

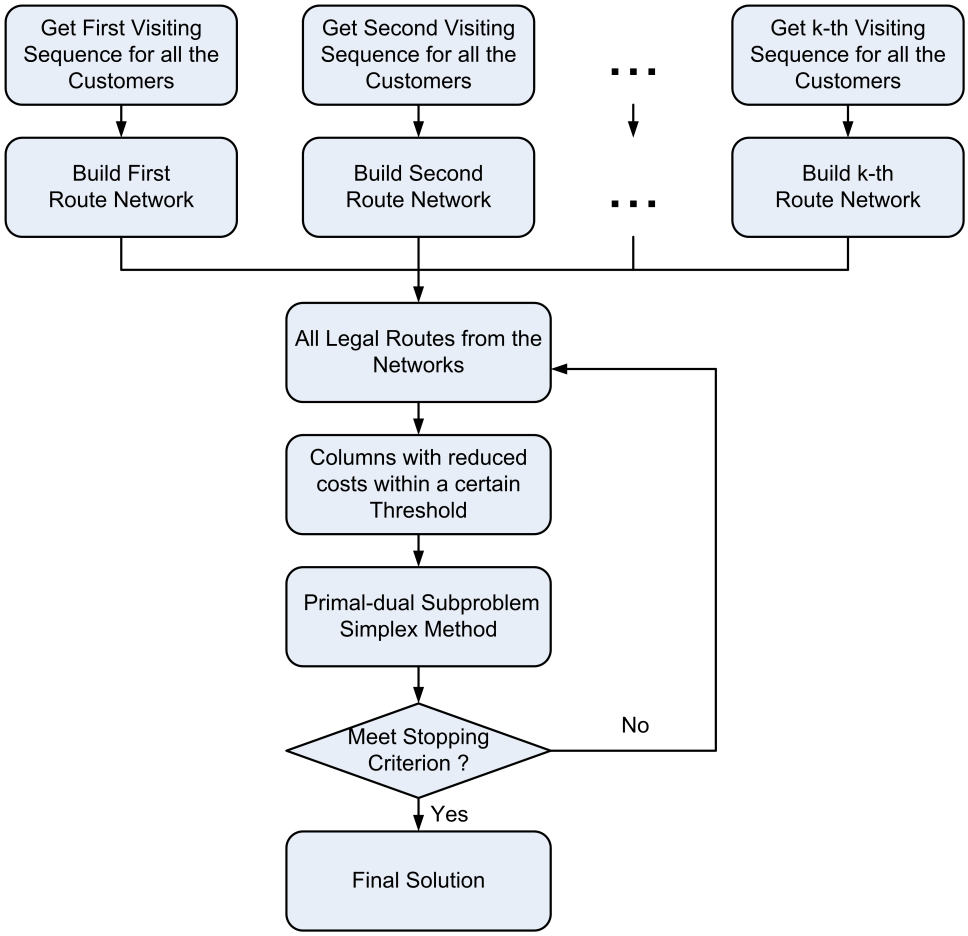

Figure 4.9: Multiple start flowchart.

As in Figure 4.9, we start from multiple visiting sequences and build the route network for each of them based on what we have discussed before. Then, during our primal-dual subproblem solution iterative process, at each iteration we get a certain number of columns from these networks based on their reduced costs, that is, within a threshold. These columns are then used in the subproblem of the primal-dual method. The iterative process continues until the stopping criterion is met, for example, optimality or a predefined early termination criterion.

4.9 Computational Results

4.9.1 Running Environment

In this study, the proposed approaches are developed using the C++ programming language. In the primal-dual loop, the Gurobi Optimization software package with academic license is used to solve the linear programming relaxation of the restricted master problem and the final MIP. The approaches and their computational runs are executed on a Windows PC with the following configuration: Windows 7 64-bit operating system with Intel Core 2 Quad CPU, 2.50GHz, RAM 8.00GB.

4.9.2 Test Problems and Results

To perform the computational testing of these solution methods, some well-known benchmark problems of the CVRP are solved, and the solutions are compared with the best solutions to date for these benchmarks. The CVRP benchmarks are available at http://www.branchandcut.org/VRP, a site maintained by T. Ralphs at Lehigh University, Bethlehem, PA. These instances are denoted as X-nY-kZ, where "X" represents the instance series, such as A, B, E, P, and so on. "Y" represents the number of nodes including the customer nodes and the depot node, and "Z" represents the number of routes.

Fukasawa et al. [52] tested their branch-and-cut-and-price algorithm with series A, B, E, F, M, and P available at www.branchandcut.org and reported optimal solutions up to 135 vertices. Based on their partial runs, they estimated that the three remaining instances from series M, with 151 to 200 vertices, could be solved in a few months of cpu time. These three remaining instances are tested using our approach.

We provide our solutions as well as best-known solutions together with their sources. The distances between any two customers used in the implementation

are rounded distances. We use TSP and VRPH to generate initial solutions together with existing known solutions to establish visiting sequences for our route networks. The solutions from our approach are post-processed using TSP and VRPH, which further feed into our approach for further improvements. Here, VRPH is a CoinOR open source project (http://www.coin-or.org/projects/VRPH.xml) that contains meta-heuristics approaches, based on a 2008 doctoral dissertation by Groër [59] as well as a recent publication [58].

Tests of M-n151-k12.vrp, M-n200-k16, and G-n262-k25

Christofides, Mingozzi, and Toth [30] introduced some test problems in 1979, as shown in Table 4.2. Of the five, researchers have worked out the optimal solutions for M-n101-k10.vrp and M-n121-k7.vrp. Many people have attempted to solve the rest three M-n151-k12.vrp, M-n200-k16.vrp, and M-n200-k17.vrp, and the best-known solutions are reported in the literature.

Table 4.2: CVRP instances for testing.

Instances (EUC_2D)	Number of Customers	Number of Vehicles	Vehicle Capacity	Tightness	Optimal Solution
M-n101-k10	100	10	200	0.91	Yes
M-n121-k7	120	7	200	0.98	Yes
M-n151-k12	150	12	200	0.93	
M-n200-k16	199	16	200	1.00	
M-n200-k17	199	17	200	0.94	
G-n262-k25	261	25	500	0.97	

Table 4.3: Improvements from existing CVRP solutions for M & G instances.

Instances (EUC_2D)	Prev Best Known	Source	Our Solution
M-n151-k12	1015	[51]	1015
M-n200-k16	1371	[61]	1289
G-n262-k25	5685	[61]	5559

For M-n151-k12.vrp, we got the same result as the best known solution in the literature. For M-n200-k16.vrp, Hasle [61] reported a solution with distance of 1371 while the total distance from our approach is 1289. The problem is illustrated in Figure 4.10, and our detailed routes are provided in Table 4.4 and Figure 4.11:

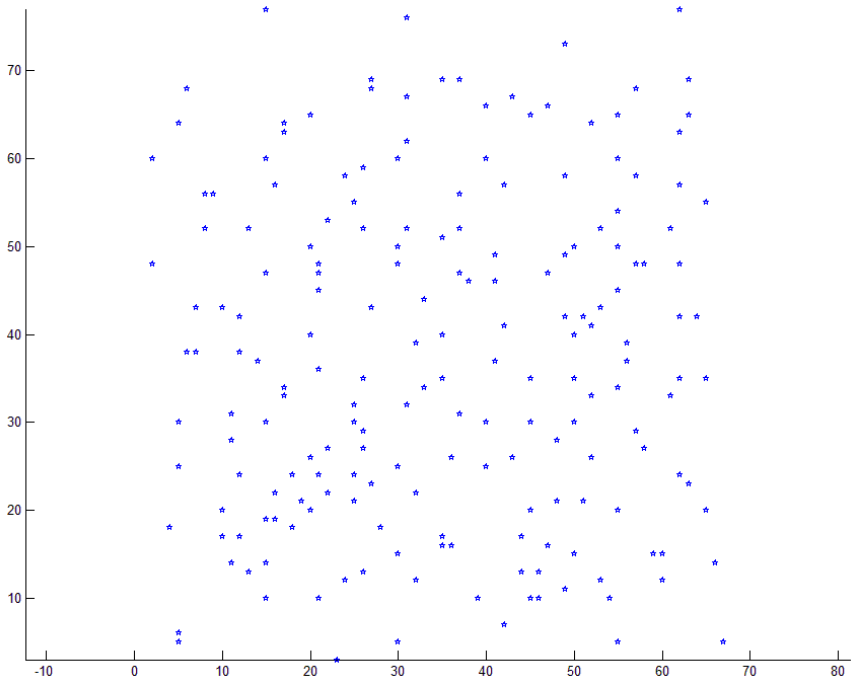

Figure 4.10: The data for M-n200-k16 instance.

We also tested our approach for problem G-n262-k25, shown in Figure 4.12. This problem was introduced by Gillett and Johnson in 1976 [54]. Hasle [61] reported a solution value of 5685 while the total route length we achieved is 5559. More details of our solution are shown in Table 4.5 and Figure 4.13. These experimental results indicate that our approach has advantages when the CVRP is large.

Table 4.4: The result for M-n200-k16.

Problem			M-n200-k16
Vehicle capacity			200
Number of customers			199
Total route length			1289
Total number of routes			16
Index	Length	Load	Route Details
1	84	199	(0 1 51 103 71 161 9 120 81 33 102 0)
2	57	200	(0 6 96 99 104 59 93 85 91 193 100 98 37 151 92 117 0)
3	66	192	(0 18 114 8 174 45 125 199 83 60 118 166 0)
4	100	200	(0 21 72 75 56 23 67 170 25 55 165 130 54 109 0)
5	72	200	(0 26 195 177 134 163 24 29 121 68 80 150 12 28 0)
6	59	199	(0 27 176 50 157 185 79 3 158 77 196 116 184 0)
7	68	199	(0 40 2 115 178 145 41 22 133 74 171 73 180 105 0)
8	75	200	(0 53 198 197 186 39 187 139 155 4 110 179 149 0)
9	40	200	(0 58 152 13 95 94 183 147 89 156 0)
10	83	200	(0 61 16 141 191 44 119 192 14 142 42 172 87 97 0)
11	96	200	(0 70 30 128 160 131 32 181 63 126 90 108 10 189 0)
12	119	199	(0 76 129 169 78 34 164 135 35 136 65 66 188 20 122 0)
13	124	200	(0 88 148 159 11 64 49 143 36 47 168 124 46 82 153 0)
14	115	198	(0 112 5 173 84 17 113 86 140 38 43 15 57 144 137 0)
15	53	200	(0 138 154 111 132 69 101 162 31 190 127 167 0)
16	78	200	(0 146 52 106 194 7 48 123 19 175 107 62 182 0)

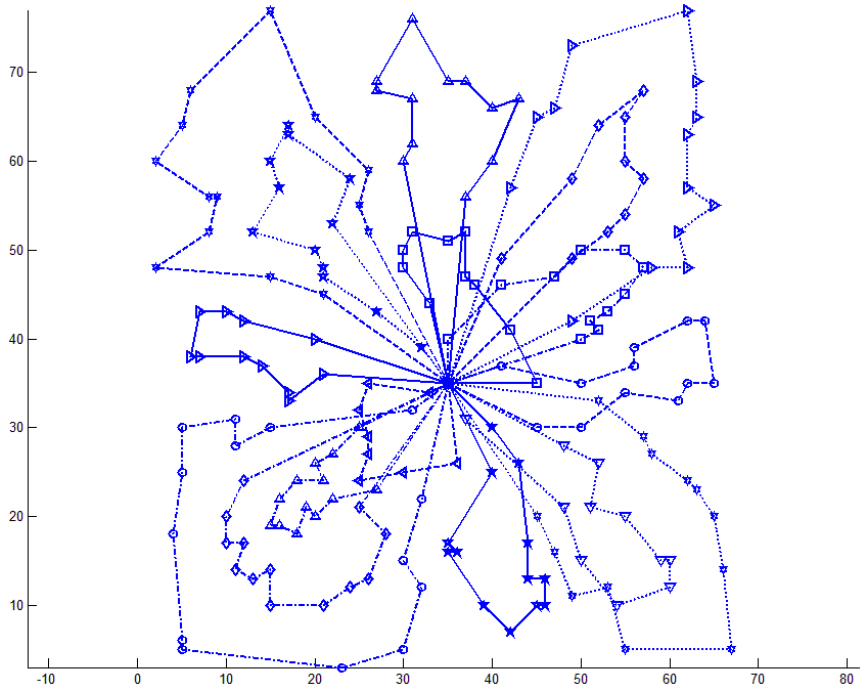

Figure 4.11: The result for M-n200-k16 instance.

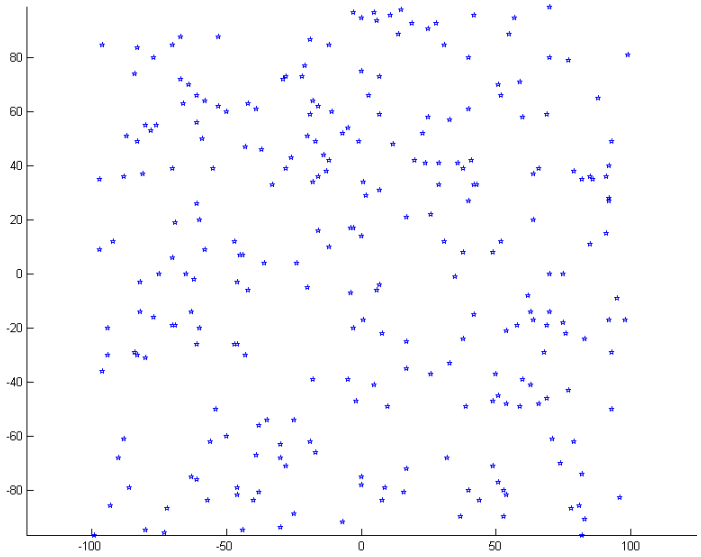

Figure 4.12: The data for G-n262-k25 instance.

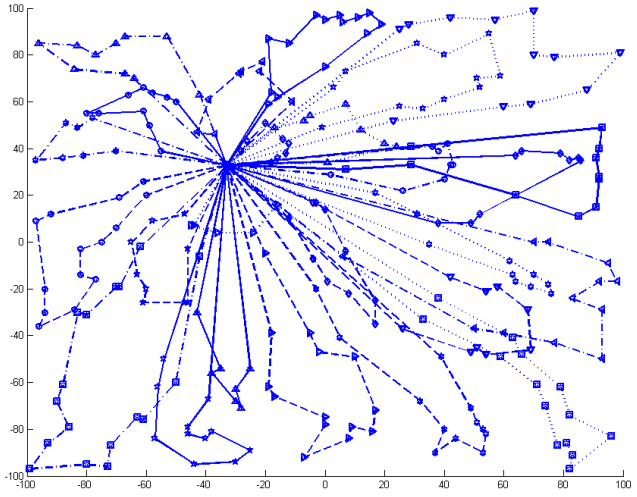

Figure 4.13: The result for G-n262-k25.

Table 4.5: The result for G-n262-k25.

Problem			G-n262-k25
Vehicle capacity			500
Number of customers			261
Total route length			5559
Total number of routes			25
Index	Length	Load	Route Details
1	176	497	(0 8 215 104 141 246 25 116 3 118 0)
2	190	473	(0 9 153 189 106 121 248 89 167 13 101 0)
3	187	491	(0 11 238 10 42 197 194 80 113 103 168 66 254 241 235 0)
4	248	498	(0 16 6 111 196 174 143 48 23 236 37 0)
5	223	497	(0 27 173 228 53 130 129 221 61 12 49 124 255 137 260 0)
6	322	497	(0 30 5 133 211 90 205 31 71 229 1 139 202 231 68 142 252 166 0)
7	300	500	(0 33 234 232 155 76 154 244 108 70 99 209 0)
8	321	474	(0 47 55 17 40 127 245 147 159 134 51 67 233 0)
9	115	496	(0 64 125 193 203 57 131 214 225 0)
10	60	485	(0 69 177 237 62 249 82 224 163 0)
11	216	500	(0 72 21 138 145 115 100 186 0)
12	88	321	(0 73 198 96 112 190 0)
13	318	498	(0 74 7 156 259 35 239 119 218 184 192 144 152 0)
14	167	486	(0 75 162 59 171 181 169 243 135 0)
15	290	499	(0 84 52 105 92 178 210 41 110 60 158 219 0)
16	234	490	(0 122 176 136 226 32 78 201 24 258 175 56 242 0)
17	276	499	(0 123 44 22 120 128 187 216 63 180 208 0)
18	346	495	(0 126 200 179 45 212 107 14 98 240 117 253 257 0)
19	253	490	(0 132 97 29 227 20 109 85 222 165 0)
20	133	499	(0 146 15 102 39 38 223 43 157 0)
21	299	485	(0 172 81 4 256 188 87 34 95 182 28 160 164 140 213 0)
22	142	499	(0 183 36 220 65 206 83 217 0)
23	127	487	(0 195 2 170 58 150 230 26 148 114 251 0)
24	378	498	(0 199 161 93 77 88 247 149 86 151 94 50 91 46 250 0)
25	150	452	(0 204 54 18 191 261 79 19 185 207 0)

4.10 Global LP Optimal Solution for CVRP

We have proposed the giant tour networks together with the primal-dual subproblem simplex method to achieve a near optimal LP solution for the CVRP. For the final primal solution x and dual solution π from the primal-dual subproblem simplex method upon convergence, x is a primal feasible solution for the CVRP, and π is a dual feasible solution for the giant tour networks but may not be a dual feasible solution for the whole CVRP network because the giant tour networks contain a subset of paths associated with the CVRP at target.

To achieve the global LP optimal solution, we need to see whether there are any paths with negative reduced costs from the CVRP network, which is an ESPPRC as we introduced earlier in this chapter.

4.10.1 The Strategy

In this section, we propose an improvement strategy to find the global LP optimal solution for the CVRP, starting from the near optimal solution that we have achieved. The main steps are provided in Algorithm 4.3.

Because the two-index formulation reviewed earlier in this chapter has an exponential number of subtour elimination constraints, we use the commodity flow formulation introduced by Ibrahim et al. [66] and Drexl and Irnich [46], which is proposed for the elementary shortest path problem. We introduced capacity constraints for solving ESPPRC. It is known that the commodity flow formulation has polynomial number constraints and provides tight LP relaxation. The full model is as follows:

Algorithm 4.3 Achieve the Global LP Optimal Solution

Step 1: Use giant tour based networks and primal-dual subproblem simplex method to achieve near optimal LP solution. Let the primal solution be x and dual solution be π, and the subproblem matrix be $\tilde{A}$, cost $\tilde{c}$.

Step 2: Update the arc costs on the CVRP network using π.

Step 3: Solve the CVRP using ESPPRC; let the resulting shortest path be p, whose traverse cost is c_p and reduced cost is $\overline{c}_p$.

Step 4: If $\overline{c}_p >= 0$, stop; we have found the global optimal solution.

Step 5: Otherwise, $\tilde{A} = \tilde{A} \cup p$, $\tilde{c} = \tilde{c} \cup c_p$, solve the new LP relaxation again,

$$\min \tilde{c}x$$

subject to

$$\tilde{A}x = b$$

$$x \geq 0$$

Let the dual solution be π; go to Step 2.

$$\min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{4.16}$$

subject to

$$\sum_{i\in V} x_{s,i} = 1 \tag{4.17}$$

$$\sum_{i\in V} x_{i,t} = 1 \tag{4.18}$$

$$\sum_{j\in V} x_{i,j} = y_i, \forall i \in V\backslash\{s,t\} \tag{4.19}$$

$$\sum_{j\in V} x_{j,i} = y_i, \forall i \in V\backslash\{s,t\} \tag{4.20}$$

$$\sum_{i\in V} d_i y_i \leq \mathcal{C} \tag{4.21}$$

$$z^k_{i,j} \leq x_{i,j}, \forall k \in K, (i,j) \in A, i \neq k, j \neq s, j \neq t \tag{4.22}$$

$$\sum_{i\in V} z^k_{s,i} = y_k, \forall k \in K \tag{4.23}$$

$$\sum_{i\in V} z^k_{i,k} = y_k, \forall k \in K \tag{4.24}$$

$$\sum_{j\in V} z^k_{i,j} = \sum_{j\in V} z^k_{j,i}, \forall k \in K, i \in V\backslash\{s,t\} \tag{4.25}$$

$$x_{i,j} = \{0,1\}, \forall (i,j) \in A \tag{4.26}$$

$$y_i = \{0,1\}, \forall i \in V\backslash\{s,t\} \tag{4.27}$$

$$z^k_{i,j} \geq 0, \forall k \in K, (i,j) \in A \tag{4.28}$$

where $x_{i,j}$ is the flow on arc $(i,j) \in A$, y_i is the flow on node $i \in V$, and $z^k_{i,j}$ is the flow on arc $(i,j) \in A$ for commodity $k \in K = V\backslash\{s,t\}$. Constraints (4.17) and (4.18) are the flow from the source and sink. Constraints (4.19) and (4.20) are flow conservation constraints. Constraints (4.21) are for resources. Constraints (4.22, 4.23, 4.24, 4.25) are subtour elimination constraints, which have a polynomial number of constraints.

Table 4.6: Global LP optimal solution for Christofides 1 instance.

Problem			Christofides 1	
Vehicle capacity			160	
Number of customers			50	
Total route length			524.611	
Total number of routes			5	
Edge weight type			EXACT 2D	
Iter	reduced cost	LP obj	length	elementary shortest path w/ resource const.
1	-14.0734	524.611	136.953	(0 11 16 21 34 30 10 45 15 44 42 19 4 0)
2	-13.5013	524.611	94.2538	(0 46 11 16 50 34 21 29 20 2 32 0)
3	-15.3513	524.611	119.569	(0 1 22 3 36 35 20 29 21 50 9 16 11 46 0)
...	...	...	...	...
28	-2.9154	522.675	93.206	(0 32 2 29 21 34 50 16 9 38 11 46 0)
29	-2.69311	521.905	83.2291	(0 11 38 9 30 34 50 16 2 32 0)
30	-3.68018	521.901	98.2796	(0 27 48 8 26 7 23 24 25 14 0)
...	...	...	...	...
76	-0.208012	520.952	122.313	(0 32 1 22 3 36 35 20 29 21 16 50 9 38 0)
77	-0.0172071	520.952	100.419	(0 5 49 10 33 45 15 44 17 4 18 0)
78	3.55×10^{-15}	520.952	116.929	(0 1 8 26 31 28 3 36 35 20 2 0)

4.10.2 A Computational Experiment

We tested our approach using an CVRP online public instance of Christofides 1 [30], in which exact Euclidean distance (Exact 2D) is used. Some computational details are listed in Table 4.6, including the corresponding LP objectives and the elementary shortest path details. In addition, some sample paths generated from the ESPPRC are illustrated in Figure 4.14. The improvement of the LP objective until reaching the global optimal is illustrated in Figure 4.15

With the primal-dual subproblem simplex method based on the giant tour network, we have the LP objective of 524.611, the final dual solution π, and subproblem of 94,839 columns. Then, we solve a series of ESPPRC problems, introduce negative reduced cost columns, and achieve a global LP optimal solution after 78 ESPPRC iterations.

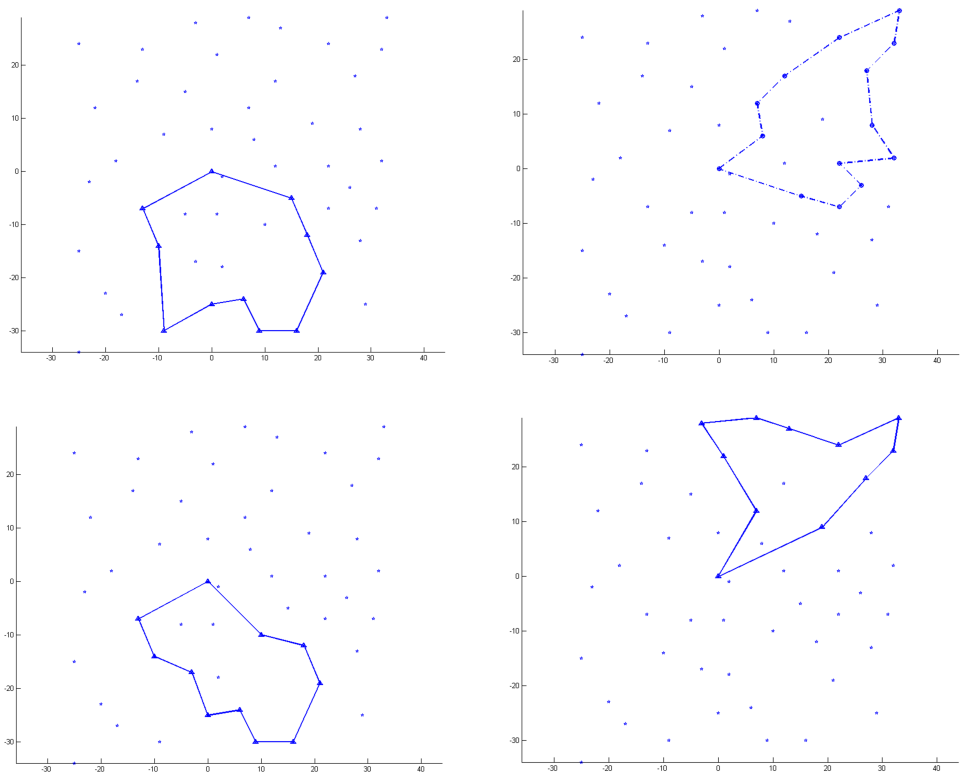

Figure 4.14: Sample paths generated from ESPPRC for Christofides 1 instance.

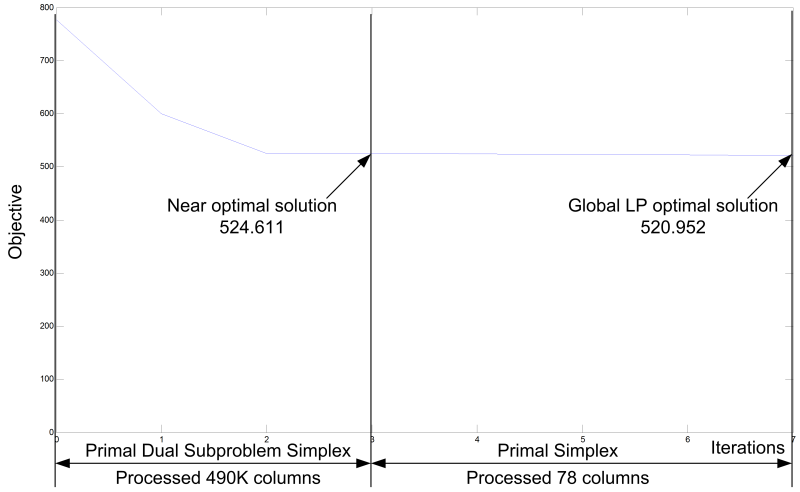

Figure 4.15: Global LP optimal solution for Christofides 1.

4.11 Capacitated Vehicle Routing Problems on Trees

The CVRP has been studied extensively throughout the years. The network representation of the underlying transportation system assumes that each node is directly connected to other nodes. Nevertheless, some practical applications may not follow this assumption, for example, some rural delivery networks in a remote area where the roads branch off from a single highway so the network exhibits a tree-like structure [24].

Some researchers presented their work for CVRP on trees in which the underlying graph is a tree. The CVRP on trees arises naturally in practical applications when the underlying transportation network has a tree structure, for example, river networks, some railway networks, pit mine railways, and mining and logging areas such as those in Northern Canada [79]. Basnet et al. [24] presented a practical scenario from the rural New Zealand dairy industry for routing milk tankers where nearly all pairs of locations are connected by unique paths, though a relatively small number of pairs of nodes have multiple paths between them in the tree-like road networks because building roads is extremely costly in the mountainous terrain and some small districts of a few farms are connected to each other through a few roads in remote areas. An example of such a network is shown in Figure 4.16.

The CVRP on trees is NP-hard, as has been shown by Labbé et al. [79]. Labbé et al. [79] presented an approach for this type of problem by getting the lower bounds based on the solutions of associated bin packing problems and the upper bounds through a linear time heuristic procedure. They then applied the branch-and-bound algorithm, taking advantage of the lower and upper bounds already obtained. Some computational results for up to 140 vertices were given. An extension of the heuristic and exact algorithms for solving homogeneous VRP on trees with duration constraints was studied by Mbaraga et al. [85], in which another exact algorithm was proposed by formulating the VRP on trees as a set covering problem and solved using column

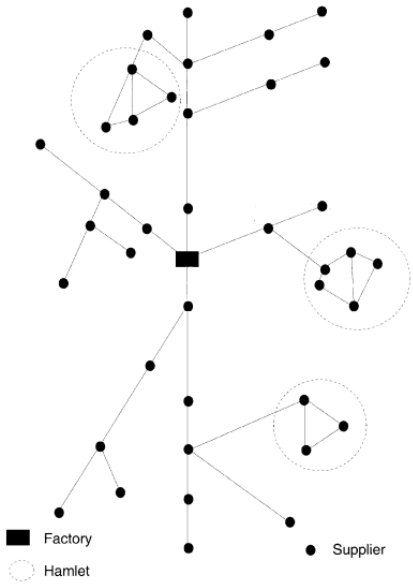

Figure 4.16: A tree-like road network of the rural New Zealand dairy industry for routing milk tankers [24].

generation. The computational results up to 140 nodes are listed with details on those solved instances. In both approaches, the computational results show that the number of successful instances decreases with increase in the number of vertices, such as 60, 90, 100, 120, and 140.

Basnet et al. [24] presented two heuristics for the VRP on tree-like road networks, which can be converted into tree structures. The first heuristic applied the general CVRP heuristic, first introduced by Clarke and Wright [32], to this CVRP on trees problem while the second started from an infeasible solution and moved to making the solution feasible. That is, first, they tried to fit the whole tree into one route, and if not possible, the tree was subdivided. This process was continued until the tree was divided into feasible routes. Furthermore, Asano et al. [11] presented an approximation algorithm for CVRP on a tree-shaped network with an assumption that the demand of a customer can be split. In addition, Katoh and Yano [72] discussed an approximation algorithm for finding good tours for pickup and delivery demands located at customer nodes in a tree-shaped network.

Chandran and Raghavan [29] presented two integer programming models, depth first ordered formulation and tree-route formulation, implemented using the AMPL model building language, with some computational results reported for 20, 40, 60, 80, and 100-node problems. The experimental data, including arc distances and customer demands, were generated using some uniform distributions.

Hasle [61] discussed how the CVRP technologies are applied in practical applications by SINTEF, a Norwegian independent multidisciplinary contract R&D organization established in 1950. Figure 4.17 shows an example of VRP problems SINTEF solved from Hasle's presentations at XVIII EWGLA in 2010. The example shows that investigation of CVRP on trees can be significant.

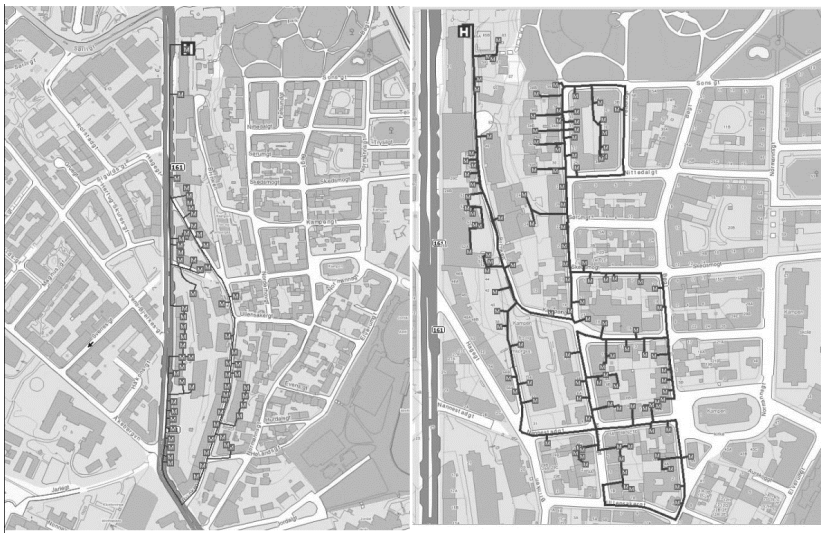

Figure 4.17: A VRP case by [61].

One feature of CVRP on trees is that the customer node may be traversed more than once, though it is serviced exactly once. Each vehicle route is a tree-like structure, and a vehicle may pass customer nodes that are served by other vehicles on the route. Other features of the CVRP on trees are the same as the CVRP, such as the objective of finding the route set that minimizes the total distance traveled; each vehicle route starting and ending at the depot; the total demand serviced by any

single vehicle not exceeding its capacity; and each customer demand being serviced by a single vehicle.

4.11.1 The Solution Method for CVRP on Trees

The solution method for CVRP on trees is similar to what we have discussed for CVRP, except that some preprocessing is needed to convert a CVRP on trees problem into a CVRP problem with a fixed node visiting sequence. To develop this solution method, we first introduce Theorem 4.11.1, which makes the CVRP on trees problem significantly easier to solve than CVRP problems.

Theorem 4.11.1. *In capacitated vehicle routing problems on trees, given a set of nodes in a vehicle's route, a minimum cost route is obtained by visiting the nodes in depth-first order.*

Proof. For the proof, please refer to [29] for detailed discussion.

□

Using the framework introduced in the previous sections of this chapter, we run a depth-first search of the CVRP tree and generate a depth-first order of the nodes on the tree. This order will be used as the node visiting sequence.

In addition, we add virtual arcs between all pairs of nodes i and j if node i is ahead of node j in the visiting sequence and i is not the parent of j. To calculate the distance of the virtual arc, we backtrack from i and j to their first common ancestor k. The virtual arc (i, j) is the case when node j is visited immediately after node i on the same vehicle route. The travel distance between i and j is the total distance between nodes i and k, plus the travel distance between nodes j and k.

Figure 4.18 shows an example of the CVRP on trees. Because node 12 is ahead of node 8, we add a virtual arc $(12, 8)$, whose distance is the sum of distance $Depot - 1 - 6 - 12$ and $Depot - 3 - 8$ because the depot is the closest common ancestor.

Suppose we want to add a virtual arc between node 12 and node 7. Both node 1 and the depot are their ancestors, but node 1 is the closest. Therefore, the distance of virtual arc $(12, 7)$ is the sum of the distance $1 - 6 - 12$ and $1 - 7$.

If node i is an ancestor of node j, then virtual arc (i, j) has the distance from node i to node j. For example, the virtual arc $(1, 12)$ has the distance of $1 - 6 - 12$.

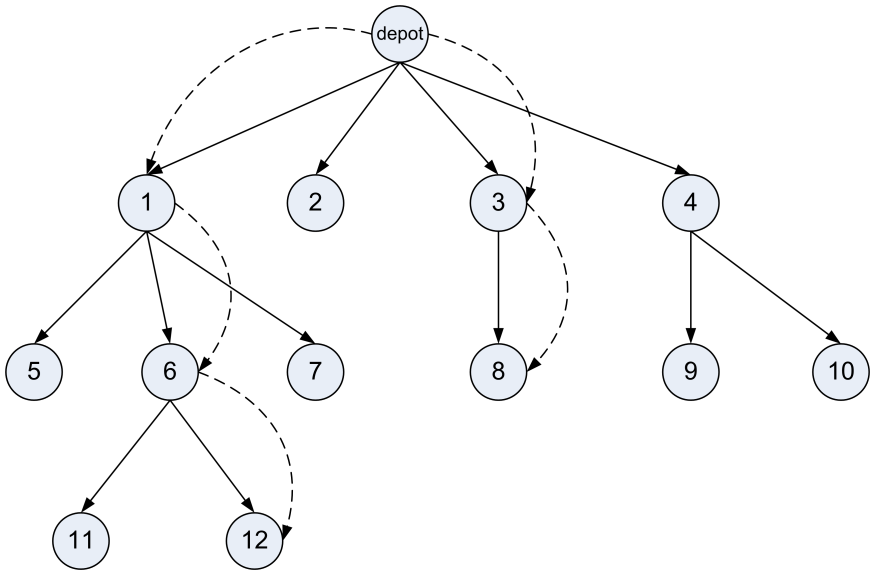

Figure 4.18: Distance between nodes on a CVRP tree.

Once we have the node visiting sequence and virtual arcs between all node pairs (i, j) if i is ahead of j in the depth-first order, we can construct the acyclic network using the same approach as that for CVRP problems, and solve this problem using the primal-dual subproblem simplex method efficiently.

4.11.2 Computational Results

It is a little bit difficult to compare the results without well-established benchmark instances. We use the data generated using uniform distribution in a similar way as introduced in [29] to do the computational experiments. Some details are provided in Table 4.7.

The vehicle capacity we set is 200, and the arc distances are integers uniformly

Table 4.7: Results for CVRP on trees.

# Customers	Capacity	Demand	Arc Distance	LP Opt	Integer Sol	Gap
140	200	[10, 90]	[1, 100]	21340.6	21739	0.0187
240	200	[10, 90]	[1, 100]	26154.2	26526	0.0142
300	200	[10, 90]	[1, 100]	74435.2	75071	0.0085
400	200	[10, 90]	[1, 100]	56306.4	57185	0.0156
500	200	[10, 90]	[1, 100]	82292.8	83652	0.0165

distributed in [1, 100]. Customer demands are integers uniformly distributed in [10, 90]. We tested different instances with the number of customers being 140, 240, 300, 400, and 500. In the table, we listed the LP optimal solutions, integer solutions, and the corresponding gaps. It can be seen that we can solve much larger CVRP on trees than those reported in the literature.

In some practical applications, there may be a few road connections that cause the corresponding network to violate the strict definition of a tree. However, if the near-tree network can be converted into the tree structure, the above method can be applied.

4.12 Summary

Our study should not be viewed as an attempt to find the optimal solutions for the CVRP problems; rather, it is an attempt to solve the CVRP problems in terms of the potential scalability and the capability of handling constraints and finding better solutions for bigger CVRP problems. The proposed capacity-expanded route network for storing a large number of routes implicitly has the good features of being acyclic and legal. Thus, we solve the problem using the set partitioning formulation and the primal-dual subproblem simplex method to find better solutions, in particular, for tighter problems. These computational results indicate that a giant tour based method can be competitive for some CVRPs.

We also extended our method to the CVRP on trees and did some computational

experiments using the data generated in a similar manner, as reported by other researchers. The result is that our approach can solve much larger problems than other methods reported in the literature.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this chapter, we make some closing remarks on our efforts in this research and highlight some interesting areas where further investigation will be worthwhile. The main contribution of this study is the two methodologies for solving airline crew pairing optimization problems and capacitated vehicle routing problems.

We have demonstrated our endeavors towards solving the large scale crew pairing problem. We proposed a duty tree approach with a compact storage scheme that shows significant savings in computer memory as shown in several computational runs. The proposed duty tree and primal-dual subproblem simplex method tailored for the duty tree work well in handling very large crew pairing problems. The results indicated that the proposed method performs very well.

Furthermore, the CVRP is the core of the VRP family and, thus, is of great interest to researchers. This present research included a promising method for solving the CVRP. We used a giant tour based concept to build the acyclic network, constructed and stored legal routes implicitly using a capacity-expanded route network, formulated a large-scale set partitioning model, and solved the problem using a primal-dual subproblem simplex method. The approach has potential scalability and can handle constraints and result in better solutions. Computational runs have shown the effectiveness of the method based on the CVRP benchmark problems. Upon convergence, we took advantage of the near optimal primal and dual solutions and used ESPPRC to achieve the LP global optimal solution. We also extended our approach to solve the CVRP on trees problem and conducted some computational experiments, which again showed that our approach is effective in solving larger problems.

5.2 Future Research

We have demonstrated some success in solving large scale airline crew pairing optimization problems and capacitated vehicle routing problems. One area that deserves further investigation is using the duty tree method to solve other airline crew pairing problems, such as the weekly and monthly problems.

The proposed method for CVRP can work with other heuristics or optimization approaches to achieve better solutions. In addition, it can be extended naturally by using a parallel computing framework because of the structure of the proposed network. In the parallel computing environment, a big number of potential routes can be included because the visiting sequences needed for a better solution may be achieved in a faster computational time.

Finally, the methods proposed in this research can be extended to many other application problems. For example, the approach for crew scheduling optimization problems could be used in the railway and bus transportation, and the proposed CVRP approach could be applied to a broad class of other VRP problems.

REFERENCES

[1] Federal Aviation Agency, www.faa.gov.

[2] Achuthan, N., Caccetta, L., and Hill, S., "An improved branch-and-cut algorithm for the capacitated vehicle routing problem," *Transportation Science*, vol. 37, pp. 153–169, 2003.

[3] Agarwal, Y., Mathur, K., and Salkin, H. M., "A set-partitioning-based exact algorithm for the vehicle routing problem," *Networks*, vol. 19, pp. 731–749, 1989.

[4] AhmadBeygi, S., Cohn, A., and Weir, M., "An integer programming approach to generating airline crew pairings," *Computers & Operations Research*, vol. 36, pp. 1284–1298, 2009.

[5] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[6] Alabas-Uslu, C. and Dengiz, B., "A self-adaptive local search algorithm for the classical vehicle routing problem," *Expert Systems with Applications*, vol. 38, pp. 8990–8998, 2011.

[7] Anbil, R., Gelman, E., Patty, B., and Tanga, R., "Recent advances in crew pairing optimization at american airlines," *Interfaces*, vol. 21, no. 1, pp. 62–74, 1991.

[8] Anbil, R., Tanga, R., and Johnson, E. L., "A global approach to crew-pairing optimization," *IBM Systems Journal*, vol. 31, pp. 71–78, 1992.

[9] Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J., *The Traveling Salesman Problem*. Princeton University Press, 2006.

[10] Arabeyre, J. P., Fearnley, J., Steiger, F. C., and Teather, W., "The airline crew scheduling problem: a survey," *Transportation Science*, vol. 3, pp. 140–163, 1969.

[11] Asano, T., Katoh, N., and Kawashima, K., "A new approximation algorithm for the capacitated vehicle routing problem on a tree," *Journal of Combinatorial Optimization*, vol. 5, pp. 213–231, 2001.

[12] Baker, E. K., Bodin, L. D., and Fisher, M., "The development of a heuristic set covering based system for aircrew scheduling," *Transportation Policy Decision Making*, no. 3, pp. 95–110, 1985.

[13] Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A., "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

[14] Baldacci, R., Christofides, N., and Mingozzi, A., "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," *Mathematical Programming, Series A*, vol. 115, pp. 351–385, 2008.

[15] Baldacci, R., Mingozzi, A., and Roberti, R., "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *European Journal of Operational Research*, vol. doi:10.1016/j.ejor.2011.07.037, 2011.

[16] Balinski, M. and Quandt, R., "On an integer program for a delivery problem," *Operations Research*, vol. 12, pp. 300–304, 1964.

[17] Ball, M. and Roberts, A., "A graph partitioning approach to airline crew scheduling," *Transportation Science*, vol. 19, pp. 95–110, 1985.

[18] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., "Branch-and-price: Column generatin for solving huge integer programs," *Operations Research*, vol. 46, pp. 316–329, 1998.

[19] Barnhart, C. and Cohn, A., "Airline scheduling planning: Accomplishments and opportunities," *Manufacturing & Service Operations Management, INFORMS*, vol. 6, no. 1, pp. 3–22, 2004.

[20] Barnhart, C., Cohn, A. M., Johnson, E. L., Klabjan, D., Nemhauser, G. L., and Vance, P. H., "Airline crew scheduling," *Handbook of Transportation Science*, vol. 56, no. Part 5, pp. 517–560, 2003.

[21] Barnhart, C., Hatay, L., and Johnson, E. L., "Deadhead selection for the long-haul crew pairing problem," *Operations Research*, vol. 43, no. 3, pp. 491–499, 1995.

[22] Barnhart, C. and Shenoi, R. G., "An approximate model and solution approach for the long-haul crew pairing problem," *Transportation Science*, vol. 32, no. 3, 1998.

[23] Barnhart, C. and Smith, B., *(Editors), Quantitative Problem Solving Methods in the Airline Industry, A Modeling Methodology Handbook*. Springer, 2012.

[24] Basnet, C., Foulds, L. R., and Wilson, J. M., "Heuristics for vehicle routing on tree-like networks," *Journal of the Operational Research Society*, vol. 50, pp. 627–635, 1999.

[25] Beasley, J. E., "Route first - cluster second methods for vehicle routing," *The International Journal of Management Science*, vol. 11, no. 4, pp. 403–408, 1983.

[26] Beasley, J. E. and Christofides, N., "An algorithm for the resource constrained shortest path problem," *Networks*, vol. 19, pp. 379–394, 1989.

[27] Berger, J. and Barkaoui, M., "A new hybrid genetic algorithm for the capacitated vehicle routing problem," *Journal of the Operational Research Society*, vol. 54, pp. 1254–1262, 2003.

[28] Bornemann, D. R., "The evolution of airline crew pairing optimization," *22nd AGIFORS Symposium Proceedings*, 1982.

[29] Chandran, B. and Raghavan, S., "Modeling and solving the capacitated vehicle routing problem on trees," *In book: The Vehicle Routing Problem: Latest Advances and New Challenges, edited by: Bruce Golden, S. Raghavan, Edward Wasil. 2008 Springer Science + Business Media, LLC*, 2008.

[30] Christofides, N., Mingozzi, A., and Toth, P., "The vehicle routing problem. in n. christofides, a. mingozzi, p. toth, and c. sandi, editors," *Combinatorial Optimization*, pp. 315–338, 1979.

[31] Chu, H. D., Gelman, E., and Johnson, E. L., "Solving large scale crew scheduling problems," *European Journal of Operations Research*, vol. 97, pp. 260–268, 1997.

[32] Clarke, G. and Wright, J. V., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, pp. 568–581, 1964.

[33] Clarke, M. and Smith, B., "The impact of operations research on the evolution of the airline industry," *AIAA Journal of Aircraft*, vol. 41, pp. 62–72, 2004.

[34] Cohn, A. and Barnhart, C., "Improving crew scheduling by incorporating key maintenance routing decisions," *Operations Research*, vol. 51, no. 3, pp. 387–396, 2003.

[35] Cook, W. J., "Concorde," *www.tsp.gatech.edu*.

[36] Cook, W. J., *In Pursuit of the Traveling Salesman, Mathematics at the Limits of Computation*. Princeton University Press, 2012.

[37] Cordeau, J.-F., Stojković, G., Soumis, F., and Desrosiers, J., "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation Science*, vol. 35, no. 4, pp. 375–388, 2001.

[38] Crainic, T. and Rousseau, J., "The column generation principle and the airline crew scheduling problem," *INFOR*, vol. 25, pp. 136–151, 1987.

[39] Crawford, B., Castro, C., and Monfroy, E., "A constructive hybrid algorithm for crew pairing optimization," *AIMSA 2006, J. Euzenat and J. Domingure(Eds.)*, pp. 45–55, 2006.

[40] Dantzig, G., Ford, L., and Fulkerson, D., "A primal-dual algorithm for linear programs," *Linear Inequalities and Related Systems, H. Kuhn and A. Tucker (editors), Princeton University Press*, pp. 171–181, 1956.

[41] Dantzig, G. B. and Ramser, J. H., "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[42] Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M. M., and Soumis, F., "Crew pairing at air france," *European Journal of Operations Research*, vol. 97, pp. 245–259, 1997.

[43] Desaulniers, G., "Managing large fixed costs in vehicle routing and crew scheduling problems solved by column generation," *Computers & Operations Research*, vol. 34, pp. 1221–1239, 2007.

[44] Desrochers, M., Desrosiers, J., and Solomon, M., "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

[45] Desrosiers, J., D., Y. Solomon, M. M., and Soumis, F., "Time constrained routing and scheduling," *Handbook in Operations Research/Management Science, Network Routing, ed. M. Ball, North-Holland, Amsterdam*, pp. 135–139, 1995.

[46] Drexl, M. and Irnich, S., "Solving elementary shortest-path problems as mixed-integer programs," tech. rep., Johannes Gutenberg University, 2012.

[47] Dror, M., "Note on the complexity of the shortest path models for column generation in vrptw," *Operations Research*, vol. 42, no. 5, pp. 977–978, 1994.

[48] Ehrgott, M. and Ryan, D. M., "Constructing robust crew schedules with bicriteria optimization," *Journal for Multi-Criteria Decision Analysis*, vol. 11, pp. 139–150, 2002.

[49] Etschmaier, M. M. and Mathaisel, D. F. X., "Airline scheduling: an overview," *Transportation Science*, vol. 19, pp. 127–138, 1985.

[50] Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C., "An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.

[51] Franceschi, R. D., Fischetti, M., and Toth, P., "A new ilp-based refinement heuristic for vehicle routing problems," *Mathematical Programming Ser. B*, vol. 105, no. 2-3, pp. 471–499, 2006.

[52] Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, F. R., "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 106, no. 3, pp. 491–511, 2006.

[53] Gerschkoff, I., "Optimizing flight crew schedules," *Interfaces*, vol. 19, pp. 29–43, 1989.

[54] Gillett, B. E. and Johnson, J. G., "Multi-terminal vehicle-dispatch algorithm," *Omega*, vol. 4, no. 6, pp. 711–718, 1976.

[55] Golden, B., Wasil, E., Kelly, J., and Chao, I.-M., "The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results, in t. crainic and g. laporte, editors," *Fleet Management and Logistics*, pp. 33–56, 1998.

[56] Gopalakrishnan, B. and Johnson, E., "Airnline crew scheduling: state-of-the-art," *Annals of Operations Research*, vol. 140, pp. 305–337, 2005.

[57] Graves, G. W., McBride, R. D., Gershkoff, I., Anderson, D. A., and Mahidhara, D., "Flight crew scheduling," *Management Science*, vol. 39, pp. 736–745, 1993.

[58] Groër, C., Golden, B., and Wasil, E., "A parallel algorithm for the vehicle routing problem," *INFORMS Journal on Computing*, vol. 23, no. 2, pp. 315–330, 2011.

[59] Groër, C., *Parallel and serial algorithms for vehicle routing problems*. Ph.D. Dissertation, University of Maryland, 2008.

[60] Hadjiconstantinou, E., Christofides, N., and Mingozzi, A., "A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations," *Annals of Operations Research*, vol. 61, pp. 21–43, 1995.

[61] Hasle, G., "Vehicle routing in practice," *XVIII EWGLA, Naples, Italy*, April 28-30 2010.

[62] Hoffman, L. H. and Padberg, M., "Solving airline crew scheduling problems by branch-and-cut," *Management Science*, vol. 39, pp. 657–682, 1993.

[63] Housos, E. and Elmroth, T., "Automatic optimization of subproblems in scheduling airline crews," *Interfaces*, vol. 27, pp. 68–77, 1997.

[64] Hu, J., *Solving linear programs using primal-dual subproblem simplex method and quasi-explicit matrices*. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Georgia, 1996.

[65] Hu, J. and Johnson, E., "Computational results with a primal-dual subproblem simplex method," *Operations Research Letters*, vol. 25, no. 4, pp. 149–157, 1999.

[66] Ibrahim, M., Maculan, N., and Minoux, M., "A strong flow-based formulation for the shortest path problem in digraphs with negative cycles," *International Transactions in Operational Research*, vol. 16, pp. 361–369, 2009.

[67] Irnich, S. and Villeneuve, D., "The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$," *GERAD G-2003-55, Montréal, Canada*, 2003.

[68] Irnich, S. and Desaulniers, G., "Shortest path problems with resource constraints," *Column Generation Chapter in GERAD 25th Anniversary Series*, pp. 33–65, 2008.

[69] Jepsen, M. K., Petersen, B., and Spoorendonk, S., "A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint," *Technical Report no. 08/01*, 2008.

[70] Johnson, E. L., "Optimization in airline scheduling: Successes, challenges, and new directions," *IMA Workshop on Travel and Transportation*, 2002.

[71] Kallehauge, B., Larsen, J., Madsen, O. B., and Solomon, M. M., "Vehicle routing problem with time windows," *Column Generation Chapter in GERAD 25th Anniversary Series*, pp. 67–98, 2008.

[72] Katoh, N. and Yano, T., "An approximation algorithm for the pickup and delivery vehicle routing problem on trees," *Discrete Applied Mathematics*, vol. 154, pp. 2335–2349, 2006.

[73] Klabjan, D., "Large-scale models in the airline industry," *In Desaulniers G, Desrosiers J., Solomon M.M. (eds), Column Generation*, pp. 163–195, 2005.

[74] Klabjan, D., Johnson, E., Nemhauser, G., Gelman, E., and Ramaswamy, S., "Airline crew scheduling with time windows and plane-count constraints," *Transportation Science*, vol. 36, no. 3, pp. 337–348, 2002.

[75] Klabjan, D., Johnson, E. L., Nemhauser, G. L., Gelman, E., and Ramaswamy, S., "Airnline crew scheduling with regularity," *Transportation Science*, vol. 35, no. 4, pp. 359–374, 2001.

[76] Klabjan, D., Johnson, E. L., Nemhauser, G. L., Gelman, E., and Ramaswamy, S., "Solving large airline crew scheduling problems: random pairing generation and strong branching," *Computational Optimization and Application*, vol. 20, pp. 73–91, 2001.

[77] Kohl, N. and Karisch, S. E., "Airline crew rostering: problem types, modeling, and optimization," *Annals of Operations Research*, vol. 127, pp. 223–257, 2004.

[78] Kolen, A., Rinnooy-Kan, A., and Trienekens, H., "Vehicle routing with time windows," *Operations Research*, vol. 35, no. 2, pp. 266–274, 1987.

[79] Labbe, M., Laporte, G., and Mercure, H., "Capacitated vehicle routing on trees," *Operations Research*, vol. 39, no. 4, pp. 616–622, 1991.

[80] Lavoie, S., Minoux, M., and Odier, E., "A new approach for crew pairing problems by column generation with application to air transportation," *European Journal of Operations Research*, vol. 35, pp. 45–58.

[81] Lettovsky, L., Johnson, E. L., and Nemhauser, G. L., "Airline crew recovery," *Transportation Science*, vol. 34, no. 4, pp. 337–348, 2000.

[82] Lysgaard, J., Letchford, N. A., and Eglese, W. R., "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, pp. 423–445, 2004.

[83] Marinakis, Y., Marinaki, M., and Dounias, G., "A hybrid particle swarm optimization algorithm for the vehicle routing problem," *Engineering Applications of Artificial Intelligence*, vol. 23, pp. 463–472, 2010.

[84] Marsten, R. E. and Shepardson, F., "Exact solution of crew problems using the set partitioning mode: recent successful applications," *Networks*, vol. 11, pp. 165–177, 1981.

[85] Mbaraga, P., Langevin, A., and Laporte, G., "Two exact algorithms for the vehicle routing problem on trees," *Naval Research Logistics*, vol. 46, pp. 75–89, 1999.

[86] Mercier, A., "A theoretical comparison of feasibility cuts for the integrated aircraft routing and crew pairing problem," *Transportation Science*, vol. 42, no. 1, pp. 87–104, 2008.

[87] Minoux, M., "Column generation techniques in combinatorial optimization: a new application to crew pairing problems," *24th AGIFORS Symposium*, pp. 15–29, 1984.

[88] Mourgaya, M. and Vanderbeck, F., "Column generation based heuristic for tactical planning in multi-period vehicle routing," *European Journal of Operational Research*, 2006.

[89] Ngueveu, S. U., Prins, C., and Calvo, R. W., "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," *Computers & Operations Research*, vol. 37, pp. 1877–1885, 2010.

[90] Papadimitriou, C. H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1998.

[91] Prins, C., "The route-first cluster-second principle in vehicle routing," *VIP 2008, Oslo, 12-14/06/2008*, 2008.

[92] Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L., "On the capacitated vehicle routing problem," *Mathematical Programming*, vol. 94, pp. 343–359, 2003.

[93] Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L., "Parallel branch and cut for capacitated vehicle routing," *Parallel Computing*, vol. 29, pp. 607–629, 2003.

[94] Reinelt, G., "Tsplib - a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, pp. 376–384, 1991.

[95] Ribeiro, G. M. and Laporte, G., "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem," *Computers & Operations Research*, vol. 39, pp. 728–735, 2012.

[96] Rochat, Y. and Taillard, E. D., "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, vol. 1, pp. 147–167, 1995.

[97] Rodríguez, A. and Ruiz, R., "A study on the effect of the asymmetry on real capacitated vehicle routing problems," *Computers & Operations Research*, vol. doi: 10.1016/j.cor.2011.10.023, 2011.

[98] Rosenberger, J. M., Schaefer, J. A., Goldsman, D., Kleywegt, A. J., Johnson, E. L., and Nemhauser, G. L., "A stochastic model of airline operations," *Transportation Science*, vol. 36, no. 4, pp. 357–377, 2002.

[99] Rubin, J., "A technique for the solution of massive set covering problems, with applications to airline crew scheduling," *Transportation Science*, vol. 7, pp. 34–48, 1973.

[100] Saddoune, M., Desaulniers, G., Elhallaoui, I., and Soumis, F., "Integrated airline crew pairing and crew assignment by dynamic constraint aggregation," *Transportation Science*, vol. Articles in advance, published online ahead of print October 28, 2011, pp. 1–17, 2011.

[101] Saddoune, M., Desaulniers, G., and Soumis, F., "Aircrew pairings with possible repetitions of the same flight number," *Computers & Operations Research*, vol. doi:10.1016/j.cor.2010.11.003, 2010.

[102] Salani, M., "Branch-and-price algorithms for vehicle routing problems," *Università Degli Studi Di Milano, http://optlab.dti.unimi.it/*, 2005.

[103] Schaefer, A. J., Johnson, E. L., Kleywegt, A. J., and Nemhauser, G. L., "Airline crew scheduling under uncertainty," *Transportation Science*, vol. 39, no. 3, pp. 340–348, 2005.

[104] Shaw, T. L., *Hybrid Column Generation for Large Network Routing Problems: with Implmentations in Airline Crew Scheduling*. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Georgia, 2003.

[105] Shebalov, S. and Klabjan, D., "Robust airline crew scheduling: move-up crews," *Transportation Science*, vol. 40, pp. 300–312, 2006.

[106] Szeto, W., Wu, Y., and Ho, S. C., "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, pp. 126–135, 2011.

[107] Tavakkoli-Moghadam, R., Safaei, N., and Gholipour, Y., "A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length," *Applied Mathematics and Computation*, vol. 176, pp. 445–454, 2006.

[108] Toth, P. and Tramontani, A., "An integer linear programming local search for capacitated vehicle routing problems," *in the book of The vehicle routing problem: latest advances and new challenges, edited by B. Golden, S. Raghavan, and E. Wasil, Springer*, 2008.

[109] Toth, P. and Vigo, D., "Models, relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics*, vol. 123, pp. 487–512, 2002.

[110] Toth, P. and Vigo, D., *The Vehicle Routing Problem*. Philadelphia : Society for Industrial and Applied Mathematics, 2002.

[111] Vance, P., Barnhart, C., Johnson, E., and Nemhauser, G., "Airline crew scheduling: A new formulation and decomposition algorithm," *Operations Research*, vol. 45, no. 2, pp. 188–200, 1997.

[112] Wedelin, D., "An algorithm for large scale 0-1 integer programming with application to airline crew scheduling," *Annals of Operations Research*, vol. 57, pp. 283–301, 1995.

[113] Weide, O., Ryan, D., and Ehrgott, M., "An iterative approach to robust and integrated aircraft routing and crew scheduling," *Computers & Operations Research*, vol. 37, pp. 833–844, 2010.

[114] West, D. B., *Introduction to Graph Theory*. Prentice Hall, Inc., 2000.

[115] Yen, J. W. and Birge, J. R., "A stochastic programming approach to the airline crew scheduling problem," *Transportation Science*, vol. 40, no. 1, pp. 3–14, 2006.